MICROSOFT™

MS™-DOS Version 2

595-3170-04

Printed in the
United States of America

ZENITH data systems

HEATH

MICROSOFT™

MS™-DOS Version 2

ZENITH data systems

HEATH

NOTICE

This software is licensed (not sold). It is licensed to sublicensees, including end-users, without either express or implied warranties of any kind on an "as is" basis.

The owner and distributors make no express or implied warranties to sublicensees, including end-users, with regard to this software, including merchantability, fitness for any purpose or non-infringement of patents, copyrights or other proprietary rights of others. Neither of them shall have any liability or responsibility to sublicensees, including end-users, for damages of any kind, including special, indirect or consequential damages, arising out of or resulting from any program, services or materials made available hereunder or the use or modification thereof.

This publication could contain technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of this publication.

Technical consultation is available for any problems you encounter in verifying the proper operation of this product. Sorry, but we are not able to evaluate or assist in the debugging of any programs you may develop. For technical assistance, write:

Zenith Data Systems Corporation
Software Consultation
Hilltop Road
St. Joseph, Michigan 49085

or call:

(616) 982-3884 Application Software/SoftStuff Products
(616) 982-3860 Operating Systems/Languages/Utilities

Consultation is available from 8:00 AM to 7:30 PM (Eastern Time Zone) on regular business days.

RESTRICTED RIGHTS LEGEND

This computer software and documentation are provided with RESTRICTED RIGHTS. Use, duplication or disclosure by the Government is subject to restrictions as set forth in the governing Rights in Technical Data and Computer Software clause — subdivision (b) (3) (B) of DAR 7-104.9(a) (May 1981) or subdivision (b) (3) (ii) of DOD FAR Supp 252.227-7013 (May 1981). Contractor/Manufacturer is Zenith Data Systems Corporation, of Hilltop Road, St. Joseph, MI 49085.

Trademarks and Copyrights

Microsoft is a registered trademark of Microsoft Corporation.
MS and the Microsoft logo are trademarks of Microsoft Corporation.
Intel is a registered trademark of Intel Corporation.
Z-DOS is a trademark of Zenith Data Systems Corporation.
Mylar is a registered trademark of DuPont Corporation.

Copyright © 1982 Microsoft Corporation, All rights reserved.
Copyright © 1984 Zenith Data Systems Corporation.

Essential requirements for using MS-DOS Version 2:

a. Distribution Media: Three 5.25-inch soft-sectored 48-tpi disks
b. Machine Configuration (minimum): Z-100 PC, 128K memory, one floppy disk drive, and CRT

ZENITH DATA SYSTEMS CORPORATION
ST. JOSEPH, MICHIGAN 49085

HEATH COMPANY
BENTON HARBOR, MICHIGAN 49022

CONTENTS

Introduction

Introduction xiii
  MS-DOS Version 2 xiii
  MS-DOS Version 1.25 xiv
Manual Organization xv
  Mandatory Reading xv
  Supplemental Reading xvii
  Specialized Reading xix
  Hardware Manuals xx
Form of Presentation xxi
  Text xxi
  Screen Displays xxi
  Entry Form Notation xxii
  Common Variables xxvi

Part I: Preparation Guide

Chapter 1 **Beginning Concepts**

Overview 1.1
Operating Systems 1.1
Disk Drives 1.3
  Drive Names 1.4
  Default Drive 1.4
  Floppy Disks 1.6
  Winchester Disks 1.9
Files 1.10
  File Names and Extensions 1.11
  File Specifications 1.13
  Reserved Device File Names 1.14
  Using Wildcard Characters to Reference Multiple Files 1.14
  Protecting Files 1.17
Commands 1.18
  Resident Commands 1.19
  Transient Commands 1.19
  Command Lines 1.20

Contents

Command Line Requirements 1.21
Special Key Entries 1.22
Initial Activities 1.23
Starting Up 1.23
Backing Up 1.24
Making Working Disks or Winchester Disk Partitions 1.24
Summary 1.25

Chapter 2 **Startup Procedure**
Purpose 2.1
Requirements 2.1
Synopsis 2.2
Startup Sequence 2.2

Chapter 3 **Backup Procedure**
Purpose 3.1
MS-DOS Distribution Disk Backup 3.1
Requirements 3.2
Distribution Backup Sequence 3.2
General Disk Backup 3.4
Requirements 3.4
DISKCOPY 3.5

Chapter 4 **Working Disk/Partition Procedures**
Purpose 4.1
Procedure Key 4.1
Working Disk Procedure 4.2
Requirements 4.2
Synopsis 4.3
Working Partition Procedure 4.11
Requirements 4.11
Synopsis 4.12

Part II: Primary Feature Guide

Chapter 5 **Command Features**
Overview 5.1

Command Types 5.1
  Resident Commands 5.2
  Transient Commands 5.4
Command Line Entry 5.5
  Parameters of a Command Line 5.6
  Command Line Requirements 5.11
Command Line Editing 5.13
  Command Line Template 5.14
  Command Line Editing Keys 5.16
  Command Line Entry Tutorial 5.18
  Control Entries 5.20
Automatic Command Entry 5.21
  The AUTOEXEC.BAT File 5.22
  Batch Processing 5.24
  Batch-Processing, Resident Commands 5.25
  Batch-Processing Tutorial 5.26
  Example Batch Files 5.32
Instructions for Single-Disk Drive Users 5.36
Command Interruption 5.38
Operating System Error Messages 5.38
Summary 5.48

Chapter 6 **Bootup Features**

Overview 6.1
Manual Bootup Command Entry Form 6.1
Preliminary Concepts 6.2
  Software/Hardware/Firmware Interaction 6.2
  Bootup Sequence 6.3
  The MS-DOS Parameters 6.4
Bootup Methods 6.5
  Automatic Bootup 6.5
  Manual Bootup 6.6
  Dynamic Bootup Parameter Defaults 6.10
  Example Manual Boot Commands 6.12
Bootup Results 6.14
Error Messages 6.15
  Error Messages Generated by the Operating System 6.16
  Error Messages Generated by the MFM-150 Monitor 6.17
Summary 6.19

Contents

Chapter 7 **Directory Features**
Overview 7.1
Directory Organization 7.2
Commands for Manipulating Directories 7.7
  Paths and Transient Commands 7.8
  Paths and Resident Commands 7.10
  Directory Access 7.11
  Directory Creation 7.13
  Directory Elimination 7.14
Example of an MS-DOS Tree Directory 7.15
File Attributes 7.19
Summary 7.19

Chapter 8 **Input/Output Features**
Overview 8.1
Input/Output Sources and Destinations 8.1
  Standard Input and Output 8.4
Input/Output Redirection 8.5
  Obtaining Input from a File or Device 8.6
  Sending Output to a File or Device 8.8
  Appending Output to an Existing File 8.10
Pipes and Filters 8.11
Printing Screen Displays 8.14
  Invoking Printer Echo 8.15
  Selectively Printing Screen Displays 8.16
Alternate Character Fonts 8.18
Summary 8.19

Chapter 9 **System Component Features**
Overview 9.1
Generic Components 9.1
  Input/Output Handler 9.2
  File Handler 9.2
  Memory Handler 9.3
  Executive 9.3
  Boot Loader 9.4
MS-DOS Components 9.4
  MSDOS.SYS Properties 9.5
  IO.SYS Properties 9.6
  COMMAND.COM Properties 9.7

MS-DOS Boot Loader Properties 9.9
Disk Locations of Components 9.9
Memory Locations of Components 9.12
Component Behavior During Bootup 9.13
Component Behavior After Bootup 9.16
Command Interpretation 9.16
Command Execution Priority 9.18
Supplementing the MSDOS.SYS Component 9.20
CONFIG.SYS Commands 9.21
CONFIG.SYS Example 9.22
CONFIG.SYS Error Message 9.23
Summary 9.23

Part III: Primary Command Guide

Chapter 10 **Command Summaries**

Overview 10.1
Alphabetical Summary 10.1
Functional Summary 10.6
Batch Processing Commands 10.7
Command Processor 10.8
Convenience Commands 10.8
Data Filtering Commands 10.9
Data/Software Analysis 10.10
Data/Software Change 10.11
Data/Software Creation 10.12
Data/Software Movement 10.13
Directory Commands 10.14
Hard Copy Commands 10.15
Manipulation Commands 10.15
Safety Commands 10.17
System Preparation Commands 10.18

Chapter 11 **Command Descriptions**

APPLY (Transient) 11.1
ASSIGN (Transient) 11.6
BACKUP (Transient) 11.13
BREAK (Resident) 11.42
CHDIR or CD (Resident) 11.44

Contents

CHKDSK (Transient) 11.58
CIPHER (Transient) 11.67
CLS (Resident) 11.74
COMMAND (Transient) 11.75
COMP (Transient) 11.83
CONFIGUR (Transient) 11.95
COPY (Resident) 11.114
CTTY (Resident) 11.126
DATE (Resident) 11.128
DEL or ERASE (Resident) 11.132
DIR (Resident) 11.137
DISKCOMP (Transient) 11.145
DISKCOPY (Transient) 11.150
ECHO (Resident, Batch-Processing) 11.156
EXE2BIN (Transient) 11.159
EXIT (Resident) 11.162
FC (Transient) 11.163
FIND (Transient) 11.176
FOR (Resident, Batch-Processing) 11.183
FORMAT (Transient) 11.186
GOTO (Resident, Batch-Processing) 11.199
IF (Resident, Batch-Processing) 11.201
MAP (Transient) 11.207
MDISK RAM-Disk Driver (Transient) 11.212
MKDIR or MD (Resident) 11.218
MODE (Transient) 11.221
MORE (Transient) 11.239
PATH (Resident) 11.241
PAUSE (Resident, Batch-Processing) 11.245
PRINT (Transient) 11.247
PROMPT (Resident) 11.258
PSC (Transient) 11.261
RDCPM (Transient) 11.263
RECOVER (Transient) 11.268
REM (Resident, Batch-Processing) 11.274
REN or RENAME (Resident) 11.277
RESTORE (Transient) 11.280
RMDIR or RD (Resident) 11.304
SEARCH (Transient) 11.306
SET (Resident) 11.309
SHIFT (Resident, Batch-Processing) 11.314
SORT (Transient) 11.318

SYS (Transient) 11.322
TIME (Resident) 11.325
TREE (Transient) 11.330
TYPE (Resident) 11.337
VER (Resident) 11.342
VERIFY (Resident) 11.343
VOL (Resident) 11.345

Part IV: Program Development Command Guide

Chapter 12 **EDLIN**
Overview 12.1
Preliminary Concepts 12.1
Invoking EDLIN 12.2
Interline Commands 12.4
Intraline Editing Functions 12.31
Error Messages 12.43

Chapter 13 **LIB**
Purpose 13.1
Entry Forms 13.1
Preliminary Concepts 13.2
Running LIB 13.6
Error Messages 13.18

Chapter 14 **LINK**
Purpose 14.1
Entry Forms 14.1
Preliminary Concepts 14.2
Running LINK 14.12
Error Messages 14.25

Chapter 15 **DEBUG**
Purpose 15.1
Entry Form 15.1
Preliminary Concepts 15.2
DEBUG Function Descriptions 15.9
Assemble 15.9
Compare 15.13
Dump 15.15

Enter 15.15
Fill 15.17
Go 15.18
Hex 15.20
Input 15.21
Load 15.22
Move 15.24
Name 15.25
Output 15.28
Quit 15.29
Register 15.30
Search 15.33
Trace 15.34
Unassemble 15.36
Write 15.39
Error Messages 15.41

Part V: Winchester Command Guide

Chapter 16 **PREP**

Purpose 16.1
Entry Form 16.1
Preliminary Concepts 16.1
Winchester Disk Features 16.2
  Platters 16.2
  Read/Write Heads 16.2
  Logical Winchester Disk Division 16.3
  Sector 16.3
  Tracks and Cylinders 16.4
  Prompts 16.4
PREP Operations 16.5
  Initializing the Disk 16.5
  Testing Media 16.6
  Initializing the Reserved Winchester Area 16.7
The Reserved Winchester Area 16.7
  The Boot Record 16.8
  Boot Indicator Byte 16.9

  The Partition Table 16.9
  The Bad Sector Table 16.11
Error Messages 16.13

Chapter 17 **PART**
Purpose 17.1
Entry Form 17.1
Preliminary Concepts 17.1
PART Operation 17.2
  Invoking PART 17.2
  The Partitioning Menu 17.5
The Partition Table 17.20
Error Messages 17.22

Chapter 18 **SHIP**
Purpose 18.1
Entry Form 18.1
Preliminary Concepts 18.1
  Prompts 18.2
  Winchester Drive Unit 18.2
  Shipping Cylinder 18.3
Error Message 18.3

Chapter 19 **DETECT**
Purpose 19.1
Entry Form 19.1
Preliminary Concepts 19.1
  Bad Sectors 19.2
  Prompts 19.3
  DETECT Operation 19.5
  DETECT Followup Activities 19.6
Error Messages 19.7

Index

Introduction

Included in this package are two versions of Microsoft's MS-DOS operating system. MS-DOS version 2 is distributed on two disks, Distribution Disk I and Distribution Disk II. MS-DOS version 2 is the version of MS-DOS that will most often be used on your microcomputer. MS-DOS version 1.25 is distributed on a single disk labeled Distribution Disk III. MS-DOS version 1.25 is included for compatibility reasons and should only be used for those rare programs that do not function properly under MS-DOS version 2.

The explanations in this manual are written primarily for MS-DOS version 2. MS-DOS version 1.25 is similar enough to MS-DOS version 2 that you can operate the features and commands of the MS-DOS version 1.25 software according to the explanations in this manual. Both versions are discussed briefly below.

MS-DOS Version 2

Your MS-DOS version 2 software package includes the version 2 operating system and transient command files.

MS-DOS version 2 is a more advanced version of the MS-DOS operating system than MS-DOS version 1.25. MS-DOS version 2 contains many new features and enhancements to make this operating system more flexible and powerful than MS-DOS version 1.25. The chapters of the manual explain each feature and command in detail.

®Microsoft is a registered trademark of Microsoft Corporation.

™MS is a trademark of Microsoft Corporation.

Introduction

MS-DOS Version 1.25

Your MS-DOS version 1.25 software package includes the version 1.25 operating system and transient command files.

This version of MS-DOS is provided for the support of application programs that do not run under MS-DOS version 2.

Some application programs might not run under MS-DOS version 2 because of the relatively large memory requirements of MS-DOS version 2. When loaded into Random Access Memory (RAM), MS-DOS version 1.25 occupies less RAM than MS-DOS version 2. Therefore, MS-DOS version 1.25 leaves more RAM free for application programs to load and operate than does MS-DOS version 2. Furthermore, MS-DOS version 2 can occupy different amounts of RAM if the CONFIG.SYS system feature is set to load nondefault shells, device drivers, or buffers. (For more information about the CONFIG.SYS feature, refer to Chapter 9, "System Component Features.") MS-DOS version 1.25 does not occupy extra RAM by loading shells, device drivers, or buffers in this way.

NOTE: You cannot access your Winchester disk with MS-DOS version 1.25.

CAUTION: Do not use commands from MS-DOS version 1.25 when running MS-DOS version 2 and do not use commands from MS-DOS version 2 while running MS-DOS version 1.25. Mixing commands from different versions of the operating system can cause damage to recorded data. Also MS-DOS version 2 will not accept the <, >, |, or \ characters in a file name because they can trigger special MS-DOS features when entered in a command line. If files you plan to use with MS-DOS version 2 have any of these characters in their file names, you should start up MS-DOS version 1.25 (from MS-DOS Distribution Disk III) and rename these files using file name characters that are valid in MS-DOS version 2. Then use the COPY command to copy the files to MS-DOS version 2 disks.

Introduction

Manual Organization

Different users of microcomputer operating systems have different levels of microcomputer experience. Because of these differences in experience, this manual has the information for different kinds of users divided into separate *parts*. Thus, you can read a minimum of documentation in order to accomplish your goals with the system.

There are some parts of the manual that all users must read, other parts that will be helpful (but not essential) to all users, and other parts that are intended only for users with special hardware or software.

Each part of this manual is divided into *chapters*. The pages of each chapter are given a unique page number that consists of the chapter number, a period, and the number of each individual page (with the sequence beginning on the first page of the chapter). For instance, page 7.9 is the ninth page of the seventh chapter. Each chapter is divided into *sections*; and most sections into *subsections*.

Mandatory Reading

The first goals after receiving the MS-DOS product are to become oriented to the product and to prepare distribution software for use in the microcomputer environment.

To accomplish these goals, you must know some general facts about operating systems. Then, after you understand the purpose of an operating system and some of its characteristics, you must perform specific activities with the microcomputer, the operating system, and other software.

Manual Organization

The following parts of the manual help you orient yourself to the operating system and to prepare your distribution software:

- Introduction (which you are now reading), and
- Part I, "Preparation Guide."

The Introduction

This introduction is mandatory reading because it explains how the MS-DOS version 2 software and documentation are presented and packaged. We recommend that all users read this introduction before beginning to use MS-DOS version 2.

The Preparation Guide

The Preparation Guide contains two different kinds of text: conceptual explanations and step-by-step procedures.

Chapter 1, "Introductory Concepts," contains explanations of all the operating system concepts you will need to know in order to prepare and begin using your software. These explanations will provide the background information necessary to perform the procedures and to begin running an application program.

Chapters 2 through 4 contain step-by-step instructions for preparing your software. Preparation of software requires that you start up the system, back up your distribution software, and arrange different software products on media that you can work with conveniently.

If the distribution software has already been backed up and arranged with other necessary software on working media, then perform only the startup procedure in Chapter 2—and skip the backup procedure in Chapter 3 and the working disk or partition procedure in Chapter 4.

NOTE: After finishing Chapters 1 through 4, most users will benefit greatly from referring frequently to the supplementary parts of the manual.

Supplemental Reading

After you have oriented yourself with the basic concepts and prepared your software, you will benefit greatly from learning more about MS-DOS features and commands.

This manual provides explanations of features and commands that can help you to make better use of the many facilities of MS-DOS version 2. Even if you had only intended to run application programs under MS-DOS version 2, reading more about it and using more of its features and commands will show you that MS-DOS version 2 is an extremely powerful and versatile operating system.

The following parts of the manual can help all users to take advantage of most MS-DOS version 2 facilities:

- Part II, "Primary Feature Guide,"
- Part III, "Primary Command Guide," and
- Quick Reference Guide.

The Primary Feature Guide

The Primary Feature Guide provides a comprehensive description of the MS-DOS version 2 features that will be beneficial to the average user.

A *feature* is any characteristic, aspect, or property of MS-DOS version 2 that is not exclusively related to a specific command. The features are explained in terms that can be understood by users who have had no prior microcomputer experience.

The system features explained in this part are related to command entry, bootup methods, disk directory structure, information transfer methods, and system structure.

The Primary Command Guide

The Primary Command Guide provides a comprehensive description of the commands in your MS-DOS software package that will be beneficial to the average user.

These commands are explained in terms that can be understood by users who have had no prior microcomputer experience.

The Primary Command Guide consists of two types of text: summaries and comprehensive descriptions.

Chapter 10, "Command Summaries," contains an alphabetical summary and a functional summary, which groups the commands under categories that describe some of their most common uses. Each summary lists the name, purpose, and entry form(s) of the primary commands.

Chapter 11, "Command Descriptions," provides a section with a comprehensive description of each primary command. The command sections are arranged in alphabetical order of command names. Most command sections describe the purpose, entry form(s), preliminary concepts, examples, and error messages of the command.

The commands that are intended for users with special hardware and/or software needs are explained in the specialized parts of the manual.

The Quick Reference Guide

The Quick Reference Guide summarizes some of the more important facts from both the Primary Feature Guide and the Primary Command Guide.

Specialized Reading

If you have special hardware or software you might need to use some MS-DOS commands and/or features that are not explained in the mandatory or supplementary reading.

The specialized reading will be beneficial to the following users:

- a user who has programming language software and wishes to develop original software,
- a user who has a Winchester disk that needs to be repartitioned or moved or purged of bad sector problems.

The following parts of the manual are written for the users listed above:

- Part IV, "Program Development Command Guide," and
- Part V, "Winchester Command Guide."

Program Development Command Guide

The Program Development Command Guide provides a comprehensive description of four commands that can be beneficial to users who have programming language software (such as a compiler or assembler) and wish to develop original software or modify acquired software.

These commands are explained in terms that can be understood by users with microcomputer assembly programming experience.

This guide explains the purpose, entry form(s), preliminary concepts, usage examples, and error messages for the commands EDLIN, LIB, LINK, and DEBUG.

The sequence of the chapters in this guide corresponds to the sequence in which these commands will probably be used.

NOTE: The EXE2BIN command (explained in Chapter 11, "Command Descriptions") can also be used in program development.

Manual Organization

Winchester Command Guide

The Winchester Command Guide provides a comprehensive description of four commands that enable you to erase all of the data on a Winchester disk, repartition the disk, change the default boot partition of the disk, prepare the disk for safe transport, or prevent access of bad sectors on the disk.

These commands are explained in terms that can be understood by users who have had no prior microcomputer experience.

This guide explains the purpose, entry form(s), preliminary concepts, examples, and error messages for the commands PREP, PART, SHIP, and DETECT (VERIFY).

The sequence of chapters in this guide corresponds to the sequence in which these commands will probably be used. However, there are circumstances in which one or more of these commands could be used alone or in a different sequence.

NOTE: The ASSIGN, BACKUP, and RESTORE commands are also especially beneficial to Winchester disk users. These commands are explained in Chapter 11, "Command Descriptions."

Hardware Manuals

This operating system manual frequently suggests that you obtain additional information by referring to your Z-100 PC series hardware documentation. This hardware documentation includes (but is not limited to) the following manuals:

- PC Series Operations Manual, and
- PC User's Guide, or
- Transportable PC User's Guide.

These manuals are provided with your microcomputer.

Form of Presentation

The material in this manual is presented in text, screen displays, or entry form notation.

Text

In explaining various features and commands of MS-DOS, this manual often presents a word within a paragraph in *italics* to indicate that the word is defined within the paragraph.

Portions of this manual will also refer to other portions of the manual where the other portions explain related topics. These cross-references usually mention the title of the applicable part, chapter, or section.

Screen Displays

Throughout this manual, many explanations of software behavior include a printed representation of a video screen display. These displays are printed in the following typestyle:

```
DISKCOPY version 2.0
```

Screen displays appear either within a paragraph or on a separate line, as shown above.

NOTE: The screen displays presented in this manual might contain different version numbers, years, and drive letters than the displays that appear when you use the software.

When a portion of a screen display is likely to be displayed differently every time the software is operated, this manual prints the variable portion in the lowercase italic form of the screen display typestyle as shown:

```
File not found: filename.ext
```

where *filename.ext* represents the information that would actually appear on the screen when a display of this kind occurs.

Entry Form Notation

Throughout this manual, many explanations instruct you to make entries by typing on the console keyboard. These entry instructions are printed in bold type and appear either within a paragraph or on a separate line.

Bold type instructions on entering a command line are used frequently in all command guides under the heading Entry Form(s). An *entry form* is a model or example of how a command line can be entered for a particular command.

The meaning of each special notation applied to entry form lines is listed in Table i.

NOTE: When it is necessary for you to press a key that is labeled with more than one character (such as the RETURN key), this manual will identify the key with all capital (uppercase) letters. The key might be labeled with both capital (uppercase) and lowercase letters on the keyboard. Therefore, a key labeled "Return" on the keyboard would be identified as "RETURN" in this manual.

Table i. Meaning of Entry Form Notations

APPEARANCE	EXAMPLE	ENTRY METHOD
On separate line; nonitalic, uppercase, bold type.	**DOTHIS**	Type the labeled keys for each character. For this example, you would press six keys.
Within a text line; nonitalic, uppercase, bold type.	**ESC**	Press the single key represented.
On separate line; italic, uppercase, bold type.	***RETURN***	Press the single key represented.
Within a text line; nonitalic, uppercase, bold type; hyphens between key names.	**CTRL-BREAK** **CTRL-ALT-DEL**	Hold down the first one or two key(s) represented and then press the last key represented. Release all keys at one time.
On separate line or within a text line; italic, lowercase, bold type.	***filespec***	This notation is a variable, which represents a certain kind of entry but will consist of different characters every time you make the entry. A list of common variables and their general definitions is provided in Table ii.

Table i (continued). Meaning of Entry Form Notations

APPEARANCE	EXAMPLE	ENTRY METHOD
Square brackets; bold type.	**[ ]**	Do not enter the square brackets. Material presented *between* square brackets is optional and should be entered only in certain cases. When a pair of square brackets is presented between another set of square brackets, the material between the inner brackets is required in more cases than the material outside the inner brackets.
Ellipses; bold type.	**...**	Do not enter ellipses. Ellipses indicate you may enter more material similar to the material preceding the ellipses.
Spaces.		Press the space bar—except when the spaces (for the sake of readability) separate key names (such as ***RETURN***) from other entries.
Punctuation marks; bold type.	**, + – ; :**	Press the punctuation key represented (with the SHIFT key, if necessary.) Do not enter square brackets or three periods for ellipses, as these marks are exceptions.

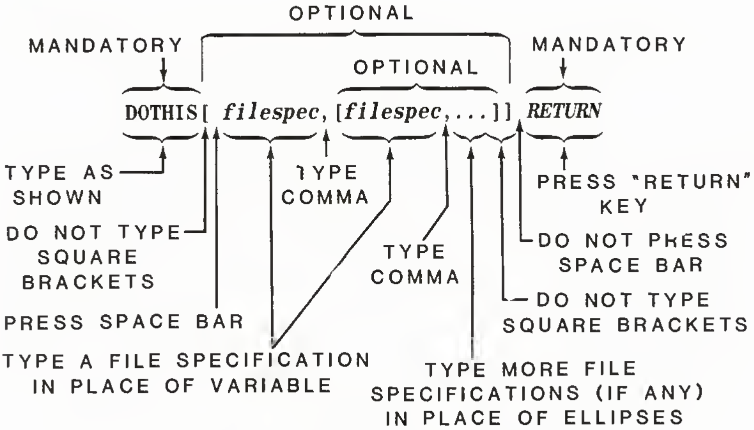

OPTIONAL
MANDATORY
OPTIONAL
MANDATORY
DOTHIS[filespec, [filespec, . . .]] RETURN
TYPE AS SHOWN
TYPE COMMA
PRESS "RETURN" KEY
DO NOT TYPE SQUARE BRACKETS
TYPE COMMA
DO NOT PRESS SPACE BAR
DO NOT TYPE SQUARE BRACKETS
PRESS SPACE BAR
TYPE A FILE SPECIFICATION IN PLACE OF VARIABLE
TYPE MORE FILE SPECIFICATIONS (IF ANY) IN PLACE OF ELLIPSES

Figure i. Reading and Entering an Entry Form Line

The sample entry form in Figure i shows you how to read and enter an entry form line from this manual.

Common Variables

Many of the entry forms used throughout this manual include strings of lowercase italic bold type to show variables. Table ii lists the general definitions for most of these variables.

The specific definition of a variable as it applies to each command is explained in detail beneath the entry form(s) in the text describing that command.

Table ii. Definitions of Common Variables

VARIABLE	DEFINITION
filename	File name consisting of primary file name (one to eight characters) and extension (zero to three characters) separated by a period in the form ***primname[.ext].*** also File name consisting of primary file name (one to eight characters with no extension) only.
filnam1 or ***filename1***	File name one, which is same as ***filename***, but used when more than one file must be specified in the same entry form.
filnam2 or ***filename2***	File name two, which is entered in the same way as ***filname1***, but usually stands for a different file than ***filname1***.
primname	Primary file name, consisting of one to eight characters that precede the optional extension, if any.
.ext	Extension of file name, preceded by period and consisting of zero to three characters.
pathname	Path name of a path to a directory or file, in the form: [\\] [***directory***] [\directory...] [\filename]

Table ii (continued). Definitions of Common Variables

VARIABLE	DEFINITION
filespec	File specification, meaning a drive name, a path name, and/or a file name in any of the following forms: [*d*:]***filename*** [*d*:]***primname***[***.ext***] [*d*:]***pathname***
d:	Drive name in most entry forms. also Destination drive name in entry forms that also contain a variable (***s***:) for the source drive.
s:	Source drive name in entry forms that also contain a variable (***d***:) for the destination drive.
/***x***	One or more switches, each consisting of a slash mark and a letter. Some switches are also followed by an argument.
p	Partition number of the desired partition on a Winchester disk.
u:	Unit number, consisting of a number from 0 through 7 (if you have eight Winchester disk units) to specify the Winchester unit you wish to use.
mm	Month. also minute.
dd	Day of the month.
yy	Year, from 1980 through 1999.
yyyy	Year, from 1980 through 2099.

Table ii (continued). Definitions of Common Variables

VARIABLE	DEFINITION
hh	Hour.
ss	Second.
cc	Hundredths of a second.
afn	Ambiguous file name, including wildcard file name characters like asterisk (*) and question mark (?) also Same as ***filespec***.
n	Numeric value, usually an integer.
dev	Device name.
cttydev	Device name of a device that allows both input and output. This device can be a console, but cannot be a printer.

Part I

Preparation Guide

Preparation Guide

This guide provides the information necessary for you to prepare to use MS-DOS on your microcomputer. This part of the manual will enable you to get your system up and running in the shortest possible time. The Preparation Guide consists of the following four chapters:

- Chapter 1, "Beginning Concepts,"
- Chapter 2, "Startup Procedures,"
- Chapter 3, "Backup Procedures," and
- Chapter 4, "Working Disk/Partition Procedures."

Chapter 1, "Beginning Concepts"—is mandatory reading for anyone not familiar with microcomputers. It provides all the conceptual information needed in order to prepare MS-DOS software for use and to run application programs. Reading this chapter will also be helpful to users who are familiar with microcomputers but unfamiliar with MS-DOS.

Chapters 2 through 4—provide you with step-by-step instructions of tasks that must be performed to prepare your MS-DOS software for use. One procedure from each of these three chapters must be performed by the first user of the MS-DOS Distribution Disks.

The final products of Chapters 2 through 4 will be one backup disk for each Distribution Disk and working disks that contain MS-DOS and application programs and/or data. By performing these procedures, you will also gain familiarity with some commonly used MS-DOS features and commands.

If you already have working disks with MS-DOS and application programs and/or data, you need to use these disks to perform only the startup procedure in Chapter 2, "Startup Procedures." Then you can skip Chapters 3 and 4 and proceed to run your application program.

NOTE: Most users will benefit greatly from also reading portions of Part II, "Primary Feature Guide" and Part III, "Primary Command Guide" as needed.

Chapter 1
Beginning Concepts

Overview

This chapter provides an introduction to concepts that are important to a full understanding of the information provided in the rest of this manual. If you are a new microcomputer user or inexperienced in operating systems, you should read this chapter as an introduction to the remainder of the manual. If you are familiar with microcomputer operating systems, you may wish to use this chapter as a review before you go on to the specific startup procedure provided in Chapter 2, "Startup Procedures."

Topics discussed in this chapter include:

- operating systems,
- disk drives,
- disks and partitions,
- files,
- commands, and
- initial activities.

The information provided is intended only to be an introduction. It will enable you to understand basic concepts and terms so that you can use the remainder of the manual with confidence.

Operating Systems

The visible part of your microcomputer hardware system consists of a central computing unit and the peripheral equipment associated and used with it. The central computing unit provides the heart of the hardware system: the microprocessor itself and memory. The central computing unit also includes control circuits for peripheral equipment and disk drives. A keyboard and screen display that provide the basic mechanism by which you use the microprocessor and communicate with it are separate components. Peripheral equipment may also include additional disk drives, a printer, or a modem. The interaction and useful operation of all this system hardware depends upon a program called an *operating system*.

Operating Systems

The operating system controls the components of your microcomputer system. It also controls the system's use and execution of subordinate *application programs*. An application program is one that is designed to perform a particular task or function but that does not directly control the hardware system functions.

Examples of application programs are word processing programs, spreadsheet programs, and accounting programs. Each application program directs the microcomputer in specific ways to perform its tasks. It is through the operating system that the link between the application program and the microcomputer system is established. Without an operating system such as MS-DOS, each application program would have to contain separate, detailed instructions for each microcomputer operation. This would make the program large and cumbersome. There would also be greater likelihood of programmer errors.

Operating systems gather together groups of detailed computer instructions and eliminate the need for them to be written into every program. The application program asks the operating system to perform a required task. The operating system then directs the microcomputer. When the application program calls on the operating system for a particular task to be completed, this request is known as a *system call*.

Thus, the operating system provides a vital link between components of your system, between your keyboard and application programs, and between your peripheral equipment and application programs.

To provide this link, MS-DOS must be transferred (or loaded) from storage media (a floppy disk or Winchester disk partition) to your microcomputer system's central computing unit. The process of loading MS-DOS into microcomputer memory is known as system *bootup*. When the central computing unit equipment is powered up, the bootup process is initiated by a small program (MFM-150 Monitor ROM) permanently stored in system hardware. This program establishes the initial communications link between MS-DOS and system hardware, enabling the transfer of the operating system to the central computing unit. Once MS-DOS has been loaded into microcomputer memory, it issues instructions and coordinates the actions of the microcomputer.

The procedures you must follow to boot up your system are described in Chapter 2, "Startup Procedures." Once bootup is successfully completed, the screen displays the *system prompt* to indicate that the operating system is ready to accept input, or commands. The system prompt you first see following bootup is the one-letter drive name of the drive from which you booted up your system followed by a right angle bracket (>). For more information on drive names and the system prompt, refer to Disk Drives in this chapter.

Information you should know to use your operating system most effectively is provided in the remainder of this chapter. This includes information on how the operating system identifies and accesses particular disk drives and disks, how the system identifies files, and how commands are input to the operating system. It also includes descriptions of the practices and procedures you should follow to protect your operating system disks and any other program or data disks you use in your system.

Disk Drives

A disk drive is a device that transfers data between your microcomputer's memory and disk storage media, whether the medium used is a 5.25-inch disk or a Winchester disk partition. (Since floppy disks and Winchester disk partitions are similar in function, both will often be referred to as "disks" in this manual. Where there is a distinction in the way the two are handled by the system, the difference will be pointed out in the text.)

Drive Names

To allow you to refer to unique disk drives and the disks and files loaded in them, MS-DOS recognizes each disk drive in the system by an alphabetic *drive name*. Drive names under MS-DOS are a single alphabetic letter, followed by a colon (:). As shipped, MS-DOS reserves particular drive names for certain hardware configurations. If your system has a single floppy drive then the drive letters A: and B: refer to this drive. If your system also has a Winchester disk then the drive letters C:, D:, E: and F: refer to the four Winchester disk partitions. If your system has two floppy drives then the drive letters A: and B: refer to the upper and lower drive slots respectively. For more information on the drive configurations supported by MS-DOS, refer to Chapter 6, "Bootup Features."

Default Drive

The system *default drive* is the drive to which the system looks to read or write information unless another drive name is specified in a given command. Initially, the default drive is the one from which MS-DOS is loaded into memory when you boot up your system. The *default system prompt* is the drive name of the default drive followed by a right angle bracket (>). For example, suppose you boot up from the upper (or only) floppy drive; the system prompt is then A> unless (or until) you specify an alternate default drive or change the system prompt using the PROMPT command (refer to Chapter 11, "Command Descriptions"). Whenever you request a program or function that requires MS-DOS to access disk storage, the operating system will automatically access the disk in drive A unless you specify an alternate drive when you enter your request.

You can change the default drive at any time by entering the drive name of the desired default drive at the system prompt and pressing RETURN. For example, if the current default drive is A and you wish to change the default to drive B, you enter **B:** and press **RETURN**. The system prompt would then be B>, showing that the default drive is drive B.

Any drive name that you specify as the default must be a valid drive in your system hardware environment. For a valid drive to be accessed, it must either contain a properly formatted disk or be assigned as a properly formatted Winchester disk partition. The disk or partition does not, however, have to contain the MS-DOS system, as it is retained in memory after the system is booted up.

All drives in your system, except the one that is the current default drive, are referred to as *non-default drives*. If drives A through H are all equipped in your system and E is the current default, then A through D and F through H are your non-default drives. If you wish to request a program that is on a disk in a non-default drive, you must precede the command or file name you enter to invoke the program with the appropriate drive name. (If you commonly use commands or programs that are not on the default drive, you may also use the PATH command to tell MS-DOS to search the non-default drives. Refer to Chapter 11 for information on the PATH command.)

While the physical construction of floppy disks and Winchester disks are different, the way in which data is written to and read from all disks is the same. Stored information, or data, is arranged in concentric rings on the surface of the disk. These rings are called *tracks*, and each one is divided into areas called *sectors*. The data in each sector is measured in units called *bytes*, each of which consists of 8 bits. A byte of data could be one letter input at the terminal keyboard, or one instruction in a program. Since a byte is such a small unit of measure, you will often see data capacity measured in kilobytes. One kilobyte (K) is equivalent to 1024 bytes.

Data is transferred to disks in the form of magnetic impulses generated by an electromagnet in the disk drive that is called the *read-write head*. As the name implies, this head can read data from the disk or write data to the disk under the control of the microcomputer and operating system. A disk drive can transfer data at any location on the disk surface almost instantaneously because the drive is normally spinning the disk at a high rate of speed.

Whenever the read-write head is instructed to read or write data at a particular location on the disk, it positions itself over the appropriate track and skims across the surface of the disk as the addressed sector spins by. Each disk has a directory that the system uses to tell the read-write head which track and sector it should access to transfer the information in the proper sequence.

The disk directory references most of the information recorded on the disk by *file name*. A file name is the name by which a group of related data (records) is identified. File names are explained later in this chapter. The characteristics of MS-DOS directories are explained in Chapter 7, "Directory Features."

Floppy Disks

A floppy disk consists of a 5.25-inch circular sheet of Mylar® plastic coated with a magnetic oxide and contained within a square plastic cover. These disks and the data stored on them are fragile. To help make sure that the disks you use and the data you store on them are not damaged, you should observe the following precautions:

- When holding the disk, touch only the disk cover. Do not touch the disk surface where it shows through the read-write access slots in the disk cover.
- Whenever the disk is not inserted in a disk drive, place it in its protective paper envelope.
- Do not allow dust, ashes, liquid, or any other foreign substance to contact the disk surface.
- Keep disks away from items such as electric motors, appliances, telephones, and so on, as these devices contain magnets that could alter the magnets impressions on the disk.

®Mylar is a Registered Trademark of the DuPont Corp.

- Do not allow disks to go through X-ray scanners such as are used in airport security systems. They may alter the magnetic impressions on the disks.

- Except for systems set for the autoboot feature, never insert a disk in a drive before turning on the drive's power, and, for all systems, *never* leave a disk in a drive when the power is being turned off. Sudden fluctuations in the power supply can cause the drive's read-write head to "crash" against the disk surface, destroying stored information.

- Do not expose disks to temperatures above 125 degrees Fahrenheit (52 Celsius) or below 40 degrees Fahrenheit (10 Celsius).

- Never press a ball-point pen or pencil directly against the cover of a disk. Instead, mark disk labels before applying them to the disk cover, or mark them using only a felt-tip pen while they are on the disk cover.

- Do not allow the disk or its cover to be bent, creased, stapled, or torn.

You can mechanically prevent the writing (or erasing) of information to (or from) your 5.25-inch floppy disks by covering the notch in the disk cover with specially-provided metallic tape tabs. When you put the tab on a disk, you have *write-protected* the disk; that is, information on the disk can only be read and cannot be altered or deleted. A 5.25-inch floppy disk with a notch that is *not* covered can be written to and erased, as well as read. Therefore, a 5.25-inch disk with an uncovered notch is *write-enabled*. Figure 1.1 illustrates this distinction between write-protected and write-enabled 5.25-inch disks.

Disk Drives

Floppy disks must be correctly inserted into the disk drives to help make sure that the data you have stored on them is correctly and efficiently accessed. First (unless your system is set for auto-boot), make sure that the disk drive is powered up *before* you insert the disk in the drive. A disk should never be left in a drive if the drive's power is not on; sudden fluctuations in the power supply can cause the read-write head to "crash" against the disk surface, destroying stored information. Then, make sure that the disk is properly oriented when you insert it in the drive.

When inserting a disk into a drive, make sure that the read-write access slot in the disk cover is toward the back of the drive. A manufacturer's label is usually affixed to one side of the disk cover, and this label should face upward as the disk is inserted in the drive. When the disk is fully inserted in the drive, fasten the drive latch.

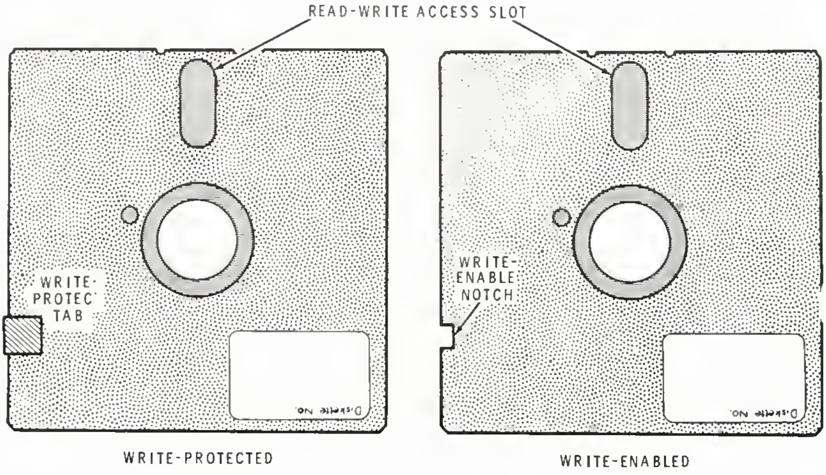


Figure 1.1. 5.25-Inch Disks

Winchester Disks

A Winchester disk consists of at least one round metal platter coated with a magnetic oxide that is permanently enclosed in a vacuum chamber within the cabinet of a microcomputer or disk drive. The Winchester disks storage capacity is far greater than that of floppy disks. To make the best use of this large capacity, the Winchester disk is divided into uniquely identifiable storage areas called *partitions*. The programs and data on a partition can be accessed in a manner similar to those stored on a floppy disk in a drive. A given partition is identified by a drive name, as described under Disk Drives in this chapter.

Winchester disk partitions are similar to floppy disks in the following ways:

- Programs and data can be accessed from a specific disk or partition by entering a command that includes the appropriate drive name for the disk or partition, if it is other than the default disk or partition.

- Each can contain an operating system, so that you can boot up by using only one partition or disk, as opposed to having to boot up with one disk or partition and using another disk for an application program, and so on.

- Different floppy disks and different Winchester partitions can contain different operating systems and still be used in the same drive.

- The storage capacity of each may be varied by using special programs (FORMAT for disks and PART for Winchester partitions) before recording data.

Winchester disk partitions and floppy disks are different in the following ways:

- The storage capacity of a Winchester disk partition is potentially much larger than that of a floppy disk.

- The storage capacity of a floppy disk is limited by the disk's physical size and by how the disk is formatted. The storage capacity of a Winchester disk partition is flexible. Winchester disk partitions can be created, enlarged, contracted, or eliminated by special commands. (The special Winchester commands including PREP and PART, are described in Part 5, "Winchester Command Guide.")

- A floppy disk can be removed from a drive and transported freely. Winchester partitions cannot be removed from the Winchester disk drive.

- MS-DOS can access all your floppy disk drives immediately upon bootup. However, MS-DOS can access a maximum of only one Winchester disk partition immediately upon bootup. Users with more than one Winchester partition must use a special command (ASSIGN) to "introduce" other partitions to the system before they can be accessed. (For more information about ASSIGN, refer to Chapter 11, "Command Descriptions.")

Refer to Chapters 16 and 17 for more information about the Winchester disk.

Files

MS-DOS enables you to create, analyze, and manipulate information by storing data in units consisting of related records called *files*. Files are stored on disks under the control of the operating system's File Manager. Files must be given names that conform to MS-DOS file naming conventions.

Whenever you wish to save or retrieve a file or issue any command that refers to a specific file, the operating system searches the appropriate disk's directory to locate the file. If necessary for command execution, the operating system makes a copy of the file's contents and stores the copy in a working area of memory until the operation you called for is completed.

File Names and Extensions

A file is identified by a name consisting of two parts: the primary file name and the extension. The *primary file name*, consisting of from one to eight characters, is always required; the *extension*, consisting of from one to three characters, is optional for many user-created files. The primary file name and extension must be separated by a period (.), as follows:

primname.ext

If you name a file with seven or fewer characters in the primary file name, MS-DOS pads the remaining "unused" characters (up to the maximum of eight) with spaces. Similarly, if you use an extension with fewer than three characters, the remaining characters are padded with spaces. File names that do not have an extension may be followed by a period, but this is not required by MS-DOS.

The characters used in primary file names and extensions can be almost any character on the keyboard except nonprinting or special function characters. Valid characters are:

A-Z	0-9	$	&	#	%
'	(	)	-	@	↑
{	}	~	`	!	_

Any lowercase letters entered as part of an MS-DOS file name or extension are converted to uppercase by the system. Thus, if you enter the file name FiLe.ExT, the system converts it to FILE.EXT.

Files

Characters that may *not* be used in either primary file names or extensions are:

? . , ; : =
* / \ + " <
>

The above invalid file name characters are reserved for particular uses within or by the operating system.

The following file name examples all conform to the file naming conventions of MS-DOS:

FORMAT.COM
4-7-83.TXT
33%-RATE.DAT
memo.doc
JOB3.HEX
ZMSGS.OVR
IO.SYS
FILE#1

While file name extensions are generally optional for user-created files, you may find it helpful to give the files you create extensions that somehow describe the contents, type, or purpose of the files. The list in Table 1.1 shows some conventions that are generally applied to the use of file name extensions.

Table 1.1. File Name Extensions

EXTENSION	FILE PURPOSE
ASM	Assembler source file
BAK	Backup file
BAS	BASIC source file
BAT	Batch file
BIN	Binary file
COB	COBOL source file
COM	Command file (binary file executable under MS-DOS)
DAT	ASCII data file
DIF	Difference file created by MS-DOS FC utiltity
DOC	ASCII document file
DVD	MS-DOS device driver file
EXE	Executable binary file that relocates when loaded
FOR	FORTRAN source file

Table 1.1 (cont.d). File Name Extensions

EXTENSION	FILE PURPOSE
LIB	Library source file
LST	ASCII list file
MAP	ASCII file of a Linked Module
OBJ	Object code from a compiler or assembler
OVR	Overlay file
REF	Expanded cross-reference file
TMP	Temporary file (VM.TMP) created by LINK
$$$	Temporary work file

File Specifications

In addition to the primary file name and extension, a given file may be referenced by the drive in which the disk that contains the file is located and/or by the directory that contains the file. When you enter a command in which a file is referenced by a drive name and/or directory path name, file name, and extension you have entered a *file specification* or *filespec*. A *filespec* may be entered in a command line in the form

d:\pathname\filename.ext

Note that when you enter a file specification, the drive name must be followed by a colon (:).

If you do not enter a drive name when you specify a file, the system will search the default disk for the specified file. Thus, if the file you wish to access is not on the disk in the default drive, you must specify the name of the appropriate drive in order for the command to be executed.

A file specification must also include a directory path name if the file you specify is not in the current directory of the default or specified disk. If you do not enter a *pathname* as part of a file specification, the system will search the current directory of the default or specified disk for the file. Refer to Chapter 7, "Directory Features," for more information on MS-DOS directories and path names.

Reserved Device File Names

Certain file names are reserved for the names of input/output devices commonly used under MS-DOS. These reserved file names are:

- CON
- PRN or LPT1
- AUX or COM1
- CLOCK$
- LPT2
- LPT3
- COM2
- NUL

Do not use these reserved file names to name a file. Even if used as part of a full file specification, these file names remain associated with their respective devices. For more information on input/output devices and reserved device file names, refer to Chapter 8, "Input/Output Features."

Using Wildcard Characters to Reference Multiple Files

Many of the commands you will enter when using MS-DOS will refer to specific files. However, there may be instances when you wish to reference more than one file, as when you want to perform the same operation for several different files. You could complete an operation for multiple files by entering a separate command for each file, but an easier way is to use special wildcard characters so that you enter the command only once and it is executed for more than one file.

Wildcard characters are variables representing one or more file name characters. The two valid MS-DOS wildcard characters are the asterisk (*) and question mark (?). The asterisk represents multiple characters in a file name; it may be used to represent an entire primary file name and or entire extension or the last characters in either a file name or extension. For example, *.* represents files with any primary file name and any extension or all files in the directory. F*.TXT would represent all files with a primary file name beginning with the letter F that have the extension .TXT. Example file names that match this specification are F.TXT, FOREWARD.TXT, FINAL.TXT, and FONT.TXT.

The question mark wildcard represents only one character in a file name, and its use is position-dependent. More than one question mark may be used in a file specification if you wish to allow for multiple variable characters. (Note that the file specification ????????.??? is equivalent to *.*.) Unlike the asterisk, the question mark may be used in the initial character positions of a primary file name or file name extension. (No specific characters may follow an asterisk wildcard in a primary file name or extension. That is, you could not use the asterisk in a file specification having the form *F.EXT.) For example, the file specification ????FEB.DAT might refer to files such as ACCTFEB.DAT, PERSFEB.DAT, LTRSFEB.DAT, and FILEFEB.DAT. Additional examples of wildcard use are provided in the paragraphs that follow. While most MS-DOS commands allow the use of wildcard characters in file specification parameters, some do not. Any restrictions applicable for a given command are described in that command's section in Chapter 11, "Command Descriptions."

Suppose you have a disk in drive B that contains the following files:

TEST1RUN.EXE
TEST2RUN.EXE
TEST3RUN.EXE
TEST1RUN.DAT
TEST2RUN.DAT
TEST3RUN.DAT
TESTALL.EXE
TEST.DAT
TEST.DOC

Files

Suppose you wished to copy all the files on the disk in drive B to a disk in drive D. You could enter a command line in the form

```
COPY B:filename D:
```

for each file. However, by using the asterisk wildcard in the source file specification, you could enter the COPY command only once and all nine files would be copied. That is, you could enter

```
COPY B:*.* D:
```

at the system prompt and press **RETURN**.

Similarly, you could use wildcards to display selective directories for the disk. If you wished to see a directory listing of all .EXE files on the disk, you could enter

```
DIR B:*.EXE
```

at the system prompt and press **RETURN**. The screen would display all files with the extension .EXE, regardless of their primary file names. (That is, a directory for files TEST1RUN.EXE, TEST2RUN.EXE, TEST3RUN.EXE, and TESTALL.EXE would be displayed.)

If you wished to display all the executable and data files (that is, files with the extensions .EXE and .DAT) for the files having names in the TEST*n*RUN series (where *n* is an integer), you could enter

```
DIR B:TEST?RUN.*
```

at the system prompt and press **RETURN**. The screen would display a directory listing for files TEST1RUN.EXE, TEST2RUN.EXE, TEST3RUN.EXE, TEST1RUN.DAT, TEST2RUN.DAT, and TEST3RUN.DAT.

To better understand the difference between the asterisk and the question mark wildcards, consider the following two command lines:

```
DIR B:TEST*.*
```
and
```
DIR B:TEST?.*
```

If you entered the first command line at the system prompt and pressed **RETURN**, the screen would display a directory of all files on the disk in drive B that have a primary file name with TEST as the first four characters of the name and any (or no) characters in the remaining character positions of the file name and extension. That is, a directory listing for all nine files would be displayed.

If you entered the second command line at the system prompt and pressed **RETURN**, the screen would display a directory of all files on the disk in drive B that have a primary file name no longer than five letters in which the first four characters are TEST, the fifth character (if it exists) is *any* character, and that have *any* extension. In this case, the screen would display a directory listing only files TEST.DOC and TEST.DAT.

Protecting Files

To make the most efficient use of your system and the programs and data you store in disk files, you should handle your disks as described earlier in this chapter. And, you should establish a regular practice or procedure for backing up your disks or copying files that are of particular importance. In addition, if you are processing information that cannot be replaced or that requires a high level of security, you should take steps to make sure that your data and programs are protected from accidental or unauthorized use, modification, or destruction.

When you make backup copies of entire disks or selected files, you should store the backup copies in a safe place separate from your working disks. Depending upon the frequency with which you use your system and the types of programs you run, you may want to back up your program and/or data disks on a daily or weekly basis. Whenever you receive new distribution disks, you should make backup copies of them immediately, even before you make your working master disks. This helps make sure that valuable program files will not be lost.

Refer to Chapter 3, "Backup Procedure," for specific steps to follow when backing up disks.

Commands

Generally speaking, a *command* is an instruction you input to your system to invoke a program (resident in the operating system or stored in a command file on disk) that can help you create, change, analyze, or move data. Commands are most often input via the keyboard at the system prompt. They may also be input to the system from batch files, as described in Chapter 5, "Command Features."

MS-DOS commands fall into two categories, as determined by the location in which the programs or routines that they call are stored: resident and transient. *Resident commands* are those used to call routines that are resident in the operating system that is stored in memory at bootup. Resident commands may be executed at any time following bootup, as the routines they call are part of COMMAND.COM. *Transient commands* are those that call a file-resident program that is not loaded into memory until requested. To execute a given transient command, the disk containing the desired program file must be in a drive that is available for use.

Knowing the location (resident or transient) of a command becomes more important as you use your system more extensively and as you use various application programs. Resident commands are available for your use with bootable application program disks that run under MS-DOS since the commands are stored in memory when you boot up your system. In addition, you may wish to copy some transient command files to application program disks to make using the program more efficient. For example, you may wish to have the MS-DOS transient command files for formatting, copying, and checking disks on your application program disks so that you may easily prepare data or documentation disks and backup disks.

Resident Commands

MS-DOS resident commands are part of the COMMAND.COM file and are always available for execution when this file is resident in memory. The COMMAND.COM file becomes resident in memory whenever you boot up your system. Thus, unlike transient commands, resident commands need not be available on disk at the time they are invoked.

The MS-DOS resident commands are listed in Chapter 5, "Command Features." For more specific information about individual commands, refer to Chapter 10, "Command Summaries," and to Chapter 11, "Command Descriptions."

Transient Commands

Transient commands are requests for functions that are performed by file-resident programs and utilities. The disk containing the program file must be available to the system at the time the function is requested so that the file's contents can be read into memory. Any file with the extension .COM or .EXE is considered a valid transient command file. Programs created with most languages have the .EXE extension and may be executed as transient command files. Transient commands can be invoked by entering the primary file name without the extension.

The MS-DOS transient commands are listed in Chapter 5, "Command Features." For more specific information about individual commands, refer to Chapter 10, "Command Summaries," and to Chapter 11, "Command Descriptions."

Command Lines

A *command line* is the entry you make at the system prompt to invoke or execute a command. The first part of a command line is always the function or command name, which may be followed by other required information identifying a file or files, to be acted upon. In addition, many MS-DOS commands support options that alter or expand the way in which the command is executed, or supply additional information to the system.

Command lines are most often entered via the keyboard and are displayed on the screen as they are typed. After the command line is entered, you must press the RETURN key to send the command to the system so that it may begin execution. (Commands may also be part of a batch file that contains a series of commands that are input to the system in sequence when the batch file is executed. For more information about batch files, refer to Chapter 5, "Command Features.")

There are certain rules that you must follow when entering command lines; more information concerning them is provided under Command Line Requirements. Generally, all MS-DOS command lines must follow the format

COMMAND ***parameters***

where **COMMAND** is the command name; and
parameters are other information regarding command options and/or user-specified parameters such as a drive name, file name, path name, or switch.

Note that, in many cases, more than one parameter may be part of a single command line. The parameters for each command are detailed in Chapter 11, "Command Descriptions."

Command Line Requirements

The following information applies to all MS-DOS commands, both resident and transient:

- The default system prompt is generated to indicate that the system is ready to accept commands and consists of the letter identifying the default drive's name followed by a right angle bracket (A>, for example).

- Commands entered at the system prompt usually are followed by one or more parameters.

- Commands and parameters may be entered in uppercase, lowercase, or a combination of both.

- Commands and their parameters must be separated by delimiters. Valid delimiters under MS-DOS are the space, comma (,), semicolon (;), equal sign (=), and TAB key.

- Delimiters other than the colon and period must not be used in a file specification (that is, enter file specifications in the form *d:filename.ext* or *d:\pathname\filename.ext*). The colon and period are the only delimiters needed in any file specification.

- When you are referring to a file name that includes an extension, you must include the file name extension as part of the command line parameters.

- Disk drives (and sometimes files) are referred to as source drives (files) and destination drives (files). A source drive (or file) is the one *from* which you will be transferring information. A destination drive (or file) is the one *to* which you will be transferring information.

- Wildcard characters (also referred to as *global file name characters*) and reserved device names, such as CON or PRN, may not be used in any command or function name.

- MS-DOS editing and control keys may be used when entering command lines. Refer to Chapter 5, "Command Features," for more information.

- For any command line that you enter, the system will not begin command execution until you press **RETURN**.

- If you want to abort or terminate execution of a command that is already being executed, you can press **CTRL-BREAK**.

- When command execution produces a large amount of output on the terminal screen, the display of that output automatically scrolls upward to show all output. You may temporarily stop the display (and thus more easily read it) by pressing **CTRL-NUMLCK**. Press any alphanumeric key (that is, any non-function or noncontrol key) to resume scrolling the display.

- Sometimes the system will prompt you to perform certain steps during command execution. When the system prompts you to `Press any key`, you can press any single alphabetic (A-Z) or numeric (0-9) key on the keyboard.

Special Key Entries

In addition to entering command lines at the system prompt, you can invoke some MS-DOS functions by pressing special function keys or by pressing a predefined sequence of keys. Some of these functions enable you to interrupt the current process, reset your system, and edit the command line entries you make at the system prompt. Many other functions are also provided.

Some of the commonly-used key sequences are known as *control key* or *control character functions* because you must first press and hold the **CTRL** key and then press at least one other key. All keys may then be released simultaneously. Two examples of control key functions are:

CTRL-ALT-DEL	resets (reboots) your system
CTRL-BREAK	interrupts and terminates the current function or operation being performed

Refer to Chapter 5 for complete information on all MS-DOS special key entries.

Initial Activities

When you receive your MS-DOS distribution disks, there are three procedures that you must complete before you begin to use your operating system: *start up* your system, *back up* your distribution software, and make *working disks* (or working partitions) that contain the operating system. These procedures are described in detail in Chapters 2 through 4 of this manual.

Starting Up

Starting up is the process of readying your hardware and software for use. You must do this before you will be able to do anything else with your operating system. Starting up includes booting up the operating system.

After startup is successfully completed, your operating system has been copied into the microcomputer, where it will remain until you reboot or turn off your system. Successful startup is evidenced by the appearance of the system prompt on the microcomputer screen.

You can start up with any disk that contains the operating system, or, in other words, with any *system disk*. A system disk is one that contains the operating system itself; it does not necessarily have to contain any of the transient commands that are provided with the operating system.

Instructions for starting up your system are provided in Chapter 2, "Startup Procedures."

Backing Up

Backing up is a process that helps assure the security of your software investment. The first thing you should do after starting up your system for the first time is back up your distribution disks. When you back up, you make a duplicate copy, or *backup disk*, of the software that is shipped to you from the factory. This makes sure that you will not have lost your MS-DOS software, should anything happen to make the original disks unusable.

Instructions on backing up your distribution disks are in Chapter 3, "Backup Procedure."

Making Working Disks or Winchester Disk Partitions

In order to increase the efficiency and convenience of your microcomputer applications, you should combine MS-DOS and your application programs so that they are on the same floppy disk or Winchester disk partition. A disk or partition that contains the operating system and application program files and/or data files is called a *working disk* or *working partition*.

Instructions for making a working disk or working partition are provided in Chapter 4, "Working Disk/Partition Procedures."

Summary

An operating system such as MS-DOS is necessary for you to efficiently use your microcomputer system. MS-DOS provides the operational link between your microcomputer hardware components and your application programs and enables you to direct your system to complete the tasks you want.

Disks drives are identified by a single-letter drive name. This enables you to access (or write) data and/or programs on a specific disk. The default drive is the drive automatically used by MS-DOS if no drive is specified for execution of a given operation.

Two types of disk storage media may be used in your microcomputer system. One type is 5.25-inch floppy disks; the other is Winchester disk partitions.

Information (data and/or programs) stored on disk are identified by file names that consist of a primary file name of up to eight characters and an optional extension of up to three characters. The file names you assign must conform to the file naming conventions of MS-DOS. Files may also be referenced by file specifications that include the file's disk and directory locations (that is, that include a drive name and a directory path name) in addition to the file name.

Instructions are given to the operating system in the form of commands entered at the system prompt or from batch files. MS-DOS commands include resident commands that are part of the operating system itself (and thus are available for execution at any time) and transient commands that are stored in disk files and not loaded into memory until requested. Some instructions are also given to MS-DOS by making special key entries.

There are three procedures you must complete before you use your operating system for the first time. You must start up your system, back up your distribution software, and make working disks or working Winchester partitions.

Chapter 2

Startup Procedure

Purpose

This Startup Procedure shows you how to begin using MS-DOS in your microcomputer. After you have successfully completed this procedure, the MS-DOS system prompt will be displayed. Then you will be able to enter commands at the system prompt.

Requirements

Before you begin this procedure, you must have the following items:

- A PC microcomputer, and
- A 5.25-inch floppy disk containing MS-DOS version 2 or MS-DOS version 1.25. (Such a disk will be referenced throughout this chapter as a *System Disk*.)

MS-DOS version 2 Distribution Disk I and MS-DOS version 1.25 Distribution Disk III are both System Disks and, therefore, suitable for startup. These are the disks you should use the first time you perform startup while preparing to back up your MS-DOS Distribution Disks.

Before you begin the startup procedure, make sure your hardware devices are connected to the proper jacks and power sources.

Synopsis

During the startup procedure, you will perform the following main activities in sequence:

Disk Insertion
Bootup
Date Entry
Time Entry

NOTE: This procedure describes how your hardware and software were set at the factory to behave during startup. As shipped, microcomputers with a Winchester disk drive are set for autoboot from the Winchester disk; microcomputers without a Winchester disk drive are set for autoboot from the (primary) floppy disk drive. If you modify your hardware or software, your system may behave differently during startup and you might have to respond in ways that are not described in this procedure.

Startup Sequence

Perform the following sequence to start up your system with MS-DOS. Note that the procedure provided here is both for systems equipped only with 5.25-inch floppy disk drives and for systems equipped with a Winchester disk drive. Therefore, all steps will not be performed for all systems. Step sequences applicable only to one type of hardware configuration are referenced where appropriate. We recommend that you read through the entire procedure before beginning, to avoid confusion.

1. Insert the System Disk into the upper (or only) floppy disk drive and close the drive.

   If you do not have a Winchester disk drive, proceed to step 4.

   If you do have a Winchester disk drive, go on to step 2.

2. If the microcomputer is off, turn on all hardware devices and then turn on the microcomputer. Press the **ESC** key to abort automatic bootup from the Winchester disk drive. Go on to step 3.

   If the microcomputer is on, press and hold the **CTRL** and **ALT** keys simultaneously. While holding these keys down, press the **DEL** key. Release all three keys at the same time. The system will reset. Immediately press the **ESC** key to abort automatic bootup from the Winchester disk drive. Go on to step 3.

3. If the monitor identification message and right arrow monitor prompt (-->) will appear in the upper left-hand corner of the screen as shown below:

```
MFM-150 Monitor, Version 0.8
Memory Size: nnnK bytes
Enter "?" for help.
-->
```

   Press **B** and **F**, and then press **RETURN** to cause your system to boot up from the System Disk in the floppy disk drive. Proceed to step 5.

   NOTE: The monitor identification message and monitor prompt are always displayed when you first turn on your system if your system has been set for manual bootup. They are not normally displayed when a system is set for automatic bootup unless:

- the boot sequence is aborted as described in the preceeding steps, or

- there is no disk in the boot drive or a nonbootable disk was placed in the boot drive.

Whenever the monitor prompt is displayed, you may enter a manual Boot command to boot up your system.

Startup Sequence

4. If the microcomputer is off, turn on all hardware devices and then turn on the microcomputer. Automatic bootup from the System Disk in the floppy disk drive will begin. Go on to step 5.

   If the microcomputer is on, press and hold the **CTRL** and **ALT** keys simultaneously. While holding these keys down, press the **DEL** key. Release all three keys at the same time. The system will reset and begin automatic bootup from the System Disk in the floppy disk drive. Go on to step 5.

   NOTE: If you system has been modified for manual bootup rather than automatic bootup, then the monitor identification message and prompt shown in step 3 will be displayed when you first turn on or reset your system. Then you must press **B** and then **RETURN** to cause your system to boot up from the System Disk in the floppy disk drive.

5. After a few moments, the MS-DOS identification message and a date prompt will be displayed in the form shown below:

```
MS-DOS version 2.xx
Copyright 1981,82,83,84 MicrocoftCorp.
Current date is Tue 1-01-1980
Enter new date:
```

6. At the Enter new date: prompt, enter the date in the form:

   mm-dd-yy

   where ***mm*** is a number from the range of 1 through 12, inclusive, designating the month;
   dd is a number from the range of 1 through 31, inclusive, representing the day of the month; and
   yy is either a two-digit numer from the range of 80 through 99, inclusive (for a year between 1980 and 1999), or a four-digit number from the range of 1980 throgh 2099, inclusive, designating the year.

   After you have entered the date, press **RETURN**.

For example, if the current (today's) date were June 11, 1984, you would enter **6-11-84** at the prompt and press **RETURN**.

After you enter the date, the screen displays

```
Current time is 8:23:15.37
Enter new time:
```

7. At the `Enter new time:` prompt, enter the time in the form

   hh[:*mm*[:*ss*]]

   where ***hh*** is a number from the range of 0 through 23, inclusive, designating the hour;
   mm is a number from the range of 0 through 59, inclusive, designating the minute; and
   ss is a number from the range of 0 through 59, inclusive, designating the second.

   If you do not enter a value for ***mm*** or for ***ss***, 00 is assumed.

   After you have entered the current time, press **RETURN**.

For example, if the current time were exactly a quarter past four in the afternoon, you would enter **16:15:00**or **16:15** and press **RETURN**.

NOTE: If your System Disk has been modified to include an AUTOEXEC.BAT file, then the date and time prompts may not be displayed.

8. After you enter a new time, MS-DOS displays the system prompt (A>) to show that you have successfully started up your system.

Startup Procedure

Startup Sequence

If your MS-DOS Distribution Disks have not yet been backed up, then proceed to Chapter 3, "Backup Procedure." When you are instructed to start up during the backup procedure, refer back to this chapter if you still need instructions about starting up.

If your MS-DOS Distribution Disks have already been backed up, then you can begin using the many features and commands of your MS-DOS software package, as described in the chapters following Chapter 4, "Working Disk/Partition Procedures."

NOTE: If you wish to start up MS-DOS in a way that is not explained in this chapter, refer to Chapter 6, "Bootup Features," for more information about the ways you can start up your system.

Chapter 3
Backup Procedure

Purpose

A backup procedure helps you to make duplicate copies (backups) of your disks. Backup disks can save you a great deal of expense and inconvenience in the event that anything happens to the disks you are using in the microcomputer.

This chapter provides both specific instructions for backing up your MS-DOS Distribution Disks and general instructions for backing up almost any other disk that has MS-DOS disk format.

Back up your MS-DOS Distribution Disks before you first use them. Then use the backup media for your future microcomputer tasks and store the original MS-DOS Distribution Disks in a safe place.

To back up your MS-DOS Distribution Disks, refer to the section on MS-DOS Distribution Disk Backup in this chapter.

To back up any other MS-DOS disk that is not copy-protected, refer to the section on General Disk Backup in this chapter.

MS-DOS Distribution Disk Backup

Your MS-DOS Distribution Disk set includes disks with two different versions of MS-DOS: version 2 and version 1.25. Therefore, you must perform the same activities one time for each version of MS-DOS.

Requirements

Before backing up your MS-DOS distribution software, you must have these items:

- A microcomputer
- MS-DOS Distribution Disk I (version 2),
- MS-DOS Distribution Disk II (version 2),
- MS-DOS Distribution Disk III (version 1.25), and
- Three blank 5.25-inch floppy disks.

CAUTION: Do not use the commands of one MS-DOS version with the system of another. After you have started up with one version, you must reset before using a command from another version.

Distribution Backup Sequence

During this procedure, you must perform the following activities in sequence:

1. Mark three adhesive paper disk labels with the titles "MS-DOS Backup Disk I—version 2," "MS-DOS Backup Disk II—version 2," and "MS-DOS Backup Disk III—version 1.25." Affix one of these labels to each of the three blank disks.

2. Start up with MS-DOS Distribution Disk I. MS-DOS version 2 will load into the microcomputer.

   NOTE: Detailed instructions on starting up are available in Chapter 2, "Startup Procedures."

MS-DOS Distribution Disk Backup

3. Begin the DISKCOPY activity by using:

   - MS-DOS Distribution Disk I as the System Disk, the Command Disk, and the first Source Disk; and
   - MS-DOS Backup Disk I as the first Backup Disk.

   NOTE: During this activity, use the version 2 DISKCOPY software and the DISKCOPY instructions of this chapter.

4. Continue the DISKCOPY activity by using:

   - MS-DOS Distribution Disk I as the System Disk and Command Disk;
   - MS-DOS Distribution Disk II as the second Source Disk; and
   - MS-DOS Backup Disk II as the second Backup Disk.

   NOTE: During this activity, use the version 2 DISKCOPY software and the DISKCOPY instructions of this chapter.

5. Start up with MS-DOS Distribution Disk III. MS-DOS version 1.25 will load into the microcomputer.

   NOTE: Detailed instructions on starting up are available in Chapter 2, "Startup Procedures."

6. Perform the DISKCOPY activity by using:

   - MS-DOS Distribution Disk III as the System Disk, the Command Disk, and the only Source Disk; and
   - MS-DOS Backup Disk III as the only Destination Disk.

   NOTE: During this activity, use the version 1.25 DISKCOPY software and the DISKCOPY instructions of this chapter.

7. Store your MS-DOS Distribution Disks away in a safe place and use your MS-DOS backup disks for future tasks.

Proceed to Chapter 4, "Working Disk/Partition Procedures."

General Disk Backup

For each disk you wish to back up, mark and affix an adhesive paper label on the blank disk to which the source information will be copied. Then start up with a System Disk as explained in Chapter 2, "Startup Procedures," and perform the DISKCOPY activity as described later in this chapter.

Requirements

Before backing up your disk(s), you must have these items:

- A microcomputer.
- An MS-DOS version 2 System Disk (such as MS-DOS Distribution Disk I) *or* an MS-DOS version 1.25 System Disk (such as MS-DOS Distribution Disk III).
- An MS-DOS version 2 Command Disk containing DISKCOPY.COM (such as MS-DOS Distribution Disk I) or an MS-DOS version 1.25 Command Disk containing DISKCOPY.COM (such as MS-DOS Distribution Disk III).
- As many source disks as you wish to duplicate.
- As many blank 5.25-inch floppy disks (backup disks) as you have source disks.

NOTE: The System Disk required here is a disk containing the version of MS-DOS listed above—with or without command files. The Command Disk required here is a formatted disk containing the specified version of the transient command, DISKCOPY.COM listed above—with or without MS-DOS. You can satisfy the requirements for both the System Disk and the Command Disk by using a single disk (such as MS-DOS Distribution Disk I or III) that has compatible versions of both MS-DOS and the required command.

CAUTION: Do not use the commands of one MS-DOS version with the system of another. After you have started up with one version, you must reset and startup before using a command from another version.

DISKCOPY

The DISKCOPY activity helps you copy all of the software from your source disk(s) to your backup disk(s). You should begin this activity with an MS-DOS Command Disk in the upper (or only) floppy disk drive and the system prompt visible on the screen.

1. Enter **DISKCOPY/V** and press **RETURN** at the `A>` system prompt. This entry invokes DISKCOPY and will cause it to verify the accuracy of its disk copying operation. DISKCOPY will first display a message and prompt in the following form:

   ```
                   DISKCOPY version 2.0

   Source drive name? (A-B) :
   ```

   NOTE: The version number and drive letters displayed by your software might differ from those shown in this example.

2. Press **A**. Then DISKCOPY will display the following prompt:

   ```
   Destination drive name? (A-B) :
   ```

3. Press **B**.

   NOTE: If you do not have a physical B drive to accommodate your backup disk, MS-DOS treats your physical A drive as a logical (or imaginary) B drive—as well as an A drive. While you run DISKCOPY, MS-DOS will often prompt you to insert the disk that belongs in drive A or the disk that belongs in drive B, depending on which is needed each time.

General Disk Backup

After you enter B for the `Destination drive name` prompt, DISKCOPY will display the following prompt:

```
Place source disk in A and destination disk in B.
Press RETURN when ready.
```

4. If you have only a single floppy drive, remove the Command Disk and insert the source disk (unless the Command Disk and the source disk are the same disk.) Then press **RETURN**. DISKCOPY will display the following prompt:

```
Formatting destination... Place disk B in drive A.
Press any key when ready.
```

and you should proceed to step 5.

If you have two floppy drives, remove the Command Disk from the upper (A) drive and insert the source disk in the upper (A) drive (unless the Command Disk and the source disk are the same disk.) Then insert a blank backup disk in the lower (B) drive and press **RETURN**. DISKCOPY will display the following prompt:

```
Formatting destination...
```

and cause the floppy disk drive light to glow for several seconds. Then proceed to step 6.

5. Remove the source disk from the drive, insert the backup disk, and press **RETURN**. DISKCOPY will cause the floppy disk drive light to glow for several seconds. Proceed to step 6.

6. Wait for DISKCOPY to begin copying.

If you have a single floppy drive, DISKCOPY will display the following message and prompt:

```
Copying... Place disk A in drive A.
Press any key when ready.
```

and you should proceed to step 7.

General Disk Backup

If you have two floppy drives, DISKCOPY will display the following message:

```
Copying...
```

and you should proceed to step 9.

7. Remove the backup disk from the drive, insert the source disk, and press **RETURN**. DISKCOPY will continue to display prompts in the following form:

   ```
   Place disk d in drive A.
   Press any key when ready.
   ```

   where *d* stands for either disk A or disk B.

8. When a prompt in this form reads `Place disk B in drive A`, insert the backup disk and press **RETURN**.

   When a prompt in this form reads `Place disk A in drive A`, insert the source disk and press **RETURN**.

9. Wait for DISKCOPY to begin verifying.

   If you have a single floppy drive, DISKCOPY will display the following message and prompt:

   ```
   Verifying... Place disk A in drive A.
   Press any key when ready.
   ```

   and you should proceed to step 10.

   If you have two floppy drives, DISKCOPY will display the following message:

   ```
   Verifying...
   ```

   and you should proceed to step 11.

General Disk Backup

10. Continue inserting and removing the source disk and backup disk as prompted until DISKCOPY displays the following prompt:

    ```
    Do you wish to copy another disk (Y/N)? <N>
    ```

    Then proceed to step 12.

11. Wait until DISKCOPY displays the following prompt:

    ```
    Do you wish to copy another disk (Y/N)? <N>
    ```

    Then proceed to step 12.

12. If you have *not* yet copied all your source disks, then press **Y** and press **RETURN**. DISKCOPY will display the `Source drive name? (A-B) :` prompt. Repeat steps 2 through 12.

    If you wish to copy source disks that contain an operating system version other than that on your System Disk, then start up with a System Disk of the appropriate version and repeat all of this DISKCOPY activity.

    If you have already made backup copies of all of your source disk media, then press **N** and press **RETURN**. MS-DOS will display the system prompt.

NOTE: If you are performing this DISKCOPY activity to back up MS-DOS distribution software, refer to the Distribution Backup Sequence section of this chapter.

Chapter 4
Working Disk/Partition Procedures

Purpose

A working disk or working partition procedure helps you to combine MS-DOS and useful application programs or data on a bootable disk or partition. Your system and application software can usually be used most efficiently when it is arranged on a working disk or working partition.

Procedure Key

This chapter contains a procedure for preparing both working disks and working partitions. You should select the proper procedure based on the circumstances described below.

- If you have one or two 5.25-inch floppy disk drives and wish to make working disks from floppy disks, then perform the Working Disk Procedure (as described within this chapter).

- If you have a Winchester disk drive, then perform the Working Partition Procedure (as described within this chapter).

- If you have a Winchester disk drive and a floppy disk drive, you can make both working partitions and working disks by performing both the Working Partition Procedure and the Working Disk Procedure (as described within this chapter).

Working Disk Procedure

Perform this procedure if you have only one or two 5.25-inch floppy disk drives and wish to make working disks from floppy disks.

Requirements

Before beginning this procedure, you must have the following items:

- A microcomputer with one or two 5.25-inch floppy disk drives.
- An MS-DOS version 2 System Disk containing FORMAT.COM (such as MS-DOS Backup Disk I) *or* an MS-DOS version 1.25 System Disk containing FORMAT.COM (such as MS-DOS Backup Disk III).
- An MS-DOS version 2 Command Disk containing CONFIGUR.COM (such as MS-DOS Backup Disk I) *or* an MS-DOS version 1.25 System Disk containing CONFIGUR.COM (such as MS-DOS Backup Disk III).
- Any number of 5.25-inch application program distribution disks and/or data disks.
- Enough blank 5.25-inch floppy disks to accommodate MS-DOS system files and application program files.

NOTE: The System Disk required here is a disk containing at least the version of MS-DOS and the command file, FORMAT.COM. The Command Disk required here is a formatted disk containing at least the specified version of the transient command file, CONFIGUR.COM—with or without MS-DOS. You can satisfy the requirements for both the System Disk and the Command Disk by using a single disk (such as MS-DOS Backup Disk I or III) that has compatible versions of both MS-DOS and the required commands.

Synopsis

During this procedure, you will perform the following activities in sequence:

Startup
FORMAT
COPY
CONFIGUR

CAUTION: You can use this procedure for making working disks with either MS-DOS version 2 or MS-DOS version 1.25. However, you must not start up with one version and then use commands from the other version.

Startup

This startup activity begins the working disk procedure for either version 2 working disks or version 1.25 working disks.

- If you are making working disks with software from your version 2 disks, then start up the system with a version 2 System Disk (such as MS-DOS Backup Disk I) in the upper (or only) drive slot.

- If you are making working disks with software from your version 1.25 disk, then start up the system with a version 1.25 System Disk (such as MS-DOS Backup Disk III) in the upper (or only) drive slot.

NOTE: Detailed instructions on starting up are available in Chapter 2, "Startup Procedures."

Proceed to the FORMAT activity.

Working Disk Procedure

FORMAT

This FORMAT activity will prepare the surfaces of your working disk(s) and copy MS-DOS to these surfaces. You should perform this activity for as many working disks as you need for both MS-DOS and your application programs or data. Begin this activity with an MS-DOS System Disk (such as MS-DOS Backup Disk I or III) in the upper (or only) floppy disk drive.

1. Mark an adhesive paper label for a blank disk with the name of the application program or data that you wish to copy. Then add "MS-DOS," the version number of the MS-DOS you are using, and the words "Working Disk" to the label. Affix the label to the disk.

2. Type the following command at the system prompt:

   ```
   FORMAT B:/S
   ```

   and press **RETURN**.

   where **B** is the drive containing the disk being prepared; and
   /S is a switch that causes the operating system to be copied.

   FORMAT will display the following message and prompt:

   ```
                 FORMAT version 2.0

   Insert new disk in drive B
   and press RETURN when ready.
   ```

   NOTE: The version number displayed by your software may differ from that shown in this manual's example.

Working Disk Procedure

3. If you have only a single floppy disk drive slot, then remove the System Disk and insert the blank working disk in the drive. Then press **RETURN**. The following prompt will be displayed:

   ```
   Place disk B in drive A:.
   Press any key when ready.
   ```

   If you have two floppy disk drive slots, then insert the blank working disk in the lower (B) drive.

4. Press **RETURN**. The light on the lower (or only) floppy disk drive will glow as FORMAT is preparing the working disk. (FORMAT might take as long as one to two minutes to prepare the disk.) Then the `System transferred` message and the following prompt will be displayed:

   ```
   Enter desired volume label (11 characters, RETURN for none)?
   ```

5. Enter 1 to 11 valid file name characters and then press **RETURN**. FORMAT will display statistics about the formatted disk and the following prompt:

   ```
   Do you wish to format another disk (Y/N)?
   ```

6. If you wish to prepare more than one working disk, then press **Y**, remove the current working disk from the drive, and proceed to step 7.

   If you have prepared all of the working disks that you desire, then proceed to step 10.

7. Wait for FORMAT to display the following prompt:

   ```
   Insert new disk in drive B
   and press RETURN when ready.
   ```

8. Mark an adhesive paper label for another blank disk, with the name of the application program or data that you wish to copy. Then add "MS-DOS," the version number of the MS-DOS you are using, and the words "Working Disk" to the label. Affix the label to the disk.

Working Disk Procedure

9. Insert this blank disk in the lower (or only) floppy disk drive and resume this activity at step 4.
10. Press **N** at the `Do you wish to format another disk (Y/N)?` prompt. MS-DOS will display the system prompt.

If you wish to put application program files and/or existing data files on your working disks immediately, then proceed to the COPY activity.

If you wish to prepare your working disks to store data that you will create in the future, then proceed to the CONFIGUR activity.

COPY

This COPY activity helps you to copy application programs and/or data files to media that has already been prepared and made bootable.

1. Insert a disk containing reliable copies of the desired application program and/or data.
2. Enter a command in the following form at the system prompt:

```
COPY primname.ext B:/V
```

and press **RETURN**.

where ***primname.ext*** stands for the file(s) that you wish to copy from the application program disk or data disk in drive A;
B is the drive that contains the working disk; and
/V is a switch that causes the accuracy of the operation to verified.

NOTE: If you want to copy all of the files from the source disk in one command, enter **COPY *.* B:/V** and press **RETURN**. The ***.*** wildcard file name stands for all the files on the source disk.

Working Disk Procedure

If you have a single floppy disk drive slot, the copying will be interrupted by prompts in the following form:

```
Place disk d in drive A:.
Press any key when ready.
```

where disk *d* can be disk A (any of the application program and/or data disks) or disk B (a working disk).

Proceed to step 3. If you have two floppy disk drive slots, then proceed to step 4.

3. Insert the application program or data disk when prompted to `Place disk A in drive A:`; and insert the Working Disk when prompted to `Place disk B in drive A:`. You might be prompted to insert these two disks alternately several times.

4. Wait until MS-DOS displays the system prompt.

5. If you wish to copy files to more working disks, then repeat steps 1 through 5.

   NOTE: You might find it useful to copy some transient command files from your MS-DOS Backup Disks to your working disk so that you have convenient access to these commands. To copy MS-DOS transient commands to your working disk, perform steps 1 through 5 by using each MS-DOS Backup Disk of the proper version as if it were an application program disk.

   If you do not wish to copy files to any more working disks, then you have finished the COPY activity.

Proceed to the CONFIGUR activity.

Working Disk Procedure

CONFIGUR

This CONFIGUR activity will adjust MS-DOS for a serial printer.

You do *not* need to perform this CONFIGUR activity if you have:

- no printer connected to your microcomputer,
- a parallel printer, or
- a printer that already runs with MS-DOS on your microcomputer.

1. If your MS-DOS is already set up for your printer, then you have completed the Working Disk Procedure.

   If your MS-DOS system is *not* yet set up for your printer, remove the working disk from the upper (or only) floppy disk drive and insert a Command Disk (such as MS-DOS Backup Disk I or III). Then enter the following command at the system prompt:

   ```
   CONFIGUR
   ```

   and press **RETURN**.

2. If you have a single floppy disk drive slot, then MS-DOS will display the `Place disk A in drive A:` prompt. Press **RETURN**. Then proceed to step 3.

   If you have two floppy disk drive slots, then proceed immediately to step 3.

3. Wait for CONFIGUR to display an identification message and a main menu, prompting you to select the kind of device you want to configure.

4. To adjust MS-DOS for your serial printer, press **A** (to Configure LPT device) at the `Enter your selection (A-C)` prompt. CONFIGUR will now prompt you to select the type of configuration.

5. Press **A** (to map parallel output to serial output). CONFIGUR will prompt you to select the parallel port to be mapped.

6. Press **A** (for the LPT1 parallel port). CONFIGUR will prompt you to select a mapping for the LPT1 parallel port.

7. Press **B** (to map to COM1). CONFIGUR will again prompt you to select the type of configuration.

8. Press **C** (to exit from this menu). CONFIGUR will again display the main CONFIGUR menu. CONFIGUR will prompt you to select the kind of device you want to configure.

9. Press **B** (to configure the COM device). CONFIGUR will prompt you to select the serial port to be configured.

10. Press **A** (for COM1). CONFIGUR will display a menu listing several hardware devices by name.

11. If your device and its characteristics are specifically listed on this menu, then press the key corresponding to the name of your hardware device.

    If your device and its characteristics are not specifically listed on this menu, then press **H** and refer to the text on CONFIGUR in Chapter 11, "Command Descriptions."

12. Press **F** (to make changes to both disk and memory). CONFIGUR will display the following prompt:

    ```
    Enter drive name with system to modify (A-B):
    ```

    NOTE: The drive names in the prompt displayed on your screen might differ from the drive names in this example.

13. Remove the disk from the upper (or only) drive and insert a working disk that you have not yet configured.

14. Press **A** (for drive A). CONFIGUR will record the selected changes on the working disk in drive A and then redisplay the main menu again.

Working Disk Procedure

15. If you wish to record the selected changes on another working disk, then remove the current working disk from the upper (or only) floppy disk drive, then repeat steps 12 through 15.

    If you do *not* wish to record the selected changes on another working disk, then press **C** (to exit from the CONFIGUR utility). MS-DOS will display the system prompt.

16. Connect your printer to the upper serial connector on the back panel of the microcomputer.

    NOTE: When your microcomputer is shipped from the factory, the left side of the back panel includes two serial connectors. One of these serial connectors is mounted above the other. The COM1 device sends output to the upper serial connector. The COM2 device sends output to the lower serial connector. The LPT1 device sends output to the one parallel connector, which is mounted on the right side of the back panel. Refer to your microcomputer hardware documentation for further information on these connectors.

17. Test whether or not the system works with the printer by simultaneously pressing the **CTRL** key and the **PRTSC** key. Then release the **CTRL** and **PRTSC** keys and press **RETURN** several times. Your printer (if properly configured and connected) will print system prompts. To discontinue this printer test, simultaneously press the **CTRL** key and the **PRTSC** key.

NOTE: If your printer does not yet work with the system or if you have hardware other than a printer (that does not work with your system), then refer to the text on CONFIGUR in Chapter 11, "Command Descriptions."

18. Test your working disk media by using it to perform a startup activity (as described in Chapter 2, "Startup Procedures").

You have completed the Working Disk Procedure.

Working Partition Procedure

Perform this Working Partition Procedure if you have a Winchester disk drive.

Requirements

Before beginning this procedure, you must have the following items:

- A microcomputer with a Winchester disk drive.
- An MS-DOS version 2 System Disk (such as MS-DOS Backup Disk I) containing version 2 FORMAT.COM.
- An MS-DOS version 2 Command Disk (such as MS-DOS Backup Disk I) containing version 2 ASSIGN.COM and CONFIGUR.COM.
- Any number of 5.25-inch application program distribution disks and/or data disks.

NOTE: The System Disk required here is a disk containing at least the version of MS-DOS listed above and the command file, FORMAT.COM. The Command Disk required here is a formatted disk containing at least the specified version of the transient command files, ASSIGN.COM and CONFIGUR.COM—with or without MS-DOS. You can satisfy the requirements for both the System Disk and the Command Disk by using a single disk (such as MS-DOS Backup Disk I) that has compatible versions of both MS-DOS and the required commands.

Working Partition Procedure

Synopsis

During this procedure, you will perform the following activities in sequence:

Floppy Disk Startup
ASSIGN
FORMAT
COPY
CONFIGUR
Partition Startup

Floppy Disk Startup

Start up the system with an MS-DOS version 2 System Disk (such as MS-DOS Backup Disk I) in the drive slot.

Detailed instructions on starting up are available in Chapter 2, "Startup Procedures."

NOTE: You cannot create a working partition with MS-DOS version 1.25. If you wish to create a working disk with MS-DOS version 1.25, then refer to the Working Disk Procedure.

Proceed to the ASSIGN activity.

ASSIGN

This ASSIGN activity helps you assign a Winchester disk partition to a drive name so the partition can be accessed. Begin this activity with an MS-DOS version 2 Command Disk (such as MS-DOS Backup Disk I) in the floppy disk drive slot.

NOTE: This activity assumes that you have retained the original partition arrangement that was shipped on your Winchester disk from the factory. This partition arrangement can be changed bv the PART command. Refer to Chapter 17, "PART," for more information on PART and partitions.

Working Partition Procedure

1. Enter the following command line at the system prompt:

   ```
   ASSIGN 0:
   ```

   and press **RETURN**.

   where ***0*** is the number of the Winchester disk drive unit.

   ASSIGN will display a table showing the numbers and types of the partitions and the drive names to which the partitions are currently assigned.

   If the display shows "Drive C=0:1" then the partition has already been assigned as drive C, proceed to the FORMAT activity. If the display shows "Drive C=Unassigned" then perform step 2.

2. Enter the following partition assignment command:

   ```
   ASSIGN 0:1 C:
   ```

   and press **RETURN**.

   where ***0***: is the unit number for your Winchester disk drive;
   1 identifies the "1. DOS" partition, which you are assigning to the drive; and
   C is the name of the drive to which you are assigning the partition.

   ASSIGN will display the following message:

   ```
   DOS partition 1 assigned to drive C
   ```

   and then MS-DOS will display the system prompt.

Proceed to the FORMAT activity.

Working Partition Procedure

FORMAT

This FORMAT activity will prepare the surface of your working partition and copy the operating system to it. Begin this activity with a version 2 System Disk (such as MS-DOS Backup Disk I) in the floppy disk drive slot.

1. Enter the following command at the system prompt:

   ```
   FORMAT C:/S
   ```

   and press **RETURN**.

   where **C** is the drive that has been assigned the MS-DOS partition; and
   /S is a switch that causes the operating system to be copied.

   FORMAT will display the following message and prompt:

   ```
                 FORMAT version 2.0

   Will FORMAT partition assigned drive C
   Press RETURN when ready.
   ```

2. Press **RETURN**. The light on the Winchester drive will glow as FORMAT prepares the partition and copies the system to it. (FORMAT may take as long as one to five minutes to prepare the partition and copy the system.) Then the `System transferred` message and the following prompt will be displayed:

   ```
   Enter desired volume label (11 characters, RETURN for none)?
   ```

3. Enter 1 to 11 valid file name characters and then press **RETURN**. FORMAT will display statistics about the formatted disk. Then MS-DOS will display the system prompt.

4. Remove the System Disk from the floppy disk drive and set it aside for a later activity.

Working Partition Procedure

If you wish to put application program files and/or existing data files on your working disks immediately, then proceed to the COPY activity.

If you wish to prepare your working disks to store data that you will create in the future, then skip ahead to the CONFIGUR activity.

COPY

This COPY activity helps you to copy application programs and/or data files to your bootable partition.

1. Insert into the floppy disk drive a disk that contains copies of the desired application program or data.

2. Enter a command in the following form at the system prompt:

   ```
   COPY primname.ext C:/V
   ```

   and press **RETURN**.

   where ***primname.ext*** stands for the file(s) that you wish to copy from the application program disk or data disk in drive A;
   C is the drive that has been assigned the partition which is receiving the application programs and/or data; and
   /V is a switch that causes the accuracy of the operation to be verified.

   NOTE: If you want to copy all of the files from the source disk at one time, enter **COPY *.* C: /V** and press **RETURN**. The *.* wildcard file name stands for all the files on the source disk.

   COPY will transfer the file(s) at a speed determined by the size of all the copied files. If you enter a wildcard file name, COPY will also display the name(s) of the file(s) copied.

Working Partition Procedure

3. Wait until MS-DOS displays the system prompt.

4. If you wish to copy files from a different application program or data disk to the working partition, then repeat steps 1 through 4.

   NOTE: You may find it useful to copy some transient command files from your MS-DOS Backup Disks to your Working Partition so that you have convenient access to these commands. To copy MS-DOS transient commands to your working partition, perform steps 1 through 4 using MS-DOS Backup Disks I and II as if they were application program disks.

   If you do not wish to copy files from a different application program or data disk to the working partition, then you are finished with the COPY activity.

Proceed to the CONFIGUR activity.

CONFIGUR

This CONFIGUR activity can help you adjust MS-DOS for a serial printer.

You do *not* need to perform this CONFIGUR activity if you have:

- no printer connected to your microcomputer,
- a parallel printer, or
- a printer that already runs with MS-DOS on your microcomputer.

1. If your MS-DOS is already adjusted for your printer, then you have completed the Winchester Working Partition Procedure.

Working Partition Procedure

If your MS-DOS system is *not* yet adjusted for your printer, insert an MS-DOS Command Disk containing CONFIGUR.COM (such as MS-DOS Backup Disk I) into the floppy disk drive (A) and enter the following command at the system prompt:

```
CONFIGUR
```

and press **RETURN**. CONFIGUR will display an identification message and a main menu, prompting you to select the kind of device you want to configure.

2. To adjust MS-DOS for your serial printer, press **A** (to Configure LPT device) at the `Enter your selection (A-F):` prompt. CONFIGUR will now prompt you to select the type of configuration.
3. Press **A** (to map parallel output to serial output). CONFIGUR will prompt you to select the parallel port to be mapped.
4. Press **A** (for the LPT1 parallel port). CONFIGUR will prompt you to select a mapping for the LPT1 parallel port.
5. Press **B** (to map to COM1). CONFIGUR will again prompt you to select the type of configuration.
6. Press **C** (to exit from this menu). CONFIGUR will again display the main CONFIGUR menu. CONFIGUR will, prompt you to select the kind of device you want to configure.
7. Press **B** (to configure the COM device). CONFIGUR will prompt you to select the serial port to be configured.
8. Press **A** (for COM1). CONFIGUR will display a menu listing several hardware devices by name.
9. If your device and its characteristics are specifically listed on this menu, then press the key corresponding to the name of your hardware device.

Working Partition Procedure

If your device and its characteristics are not specifically listed on this menu, then press **H** and refer to the text on CONFIGUR in Chapter 11, "Command Descriptions."

10. Press **F** (to make changes to both disk and memory). CONFIGUR will display the following prompt:

    ```
    Enter drive name with system to modify (A-F):
    ```

11. Press **C** (for drive C, which has been assigned the "1. DOS" partition.) CONFIGUR will record the changes you selected on the working partition assigned to drive C and then redisplay the main menu again.

12. Press **C** (to exit from the CONFIGUR utility). MS-DOS will display the system prompt.

13. Connect your printer to the upper serial connector on the back panel of the microcomputer.

    NOTE: When your microcomputer is shipped from the factory, the left side of the back panel includes two serial connectors. One of these serial connectors is mounted above the other. The COM1 device sends output to the upper serial connector. The COM2 device sends output to the lower serial connector. The LPT1 device sends output to the one parallel connector, which is mounted on the right side of the back panel. Refer to your microcomputer hardware documentation for further information on these connectors.

14. Test whether or not the system works with the printer by simultaneously pressing the **CTRL** key and the **PRTSC** key. Then release the CTRL and PRTSC keys and press **RETURN** several times. Your printer (if properly configured and connected) will print system prompts. To discontinue this printer test, simultaneously press the **CTRL** key and the **PRTSC** key.

    NOTE: If your printer does not yet work with the system or if you have hardware other than a printer (that does not work with your system), then refer to the text on CONFIGUR in Chapter 11, "Command Descriptions."

Proceed to the Partition Startup activity.

Partition Startup

This activity will help you to start up your working partition.

1. Reset your system by pressing and holding the **CTRL** and **ALT** keys simultaneously, and then pressing the **DEL** key. Release all three keys at the same time.

2. If you have not changed any hardware switch settings since you received your system, automatic bootup from the Winchester disk drive will begin. Proceed to step 3.

   If the MFM-150 Monitor identification message and right arrow monitor prompt (-->) are displayed, then your hardware has been set for manual bootup. To boot from your working partition, enter

   ```
   BW0:1
   ```

   at the monitor prompt and press **RETURN**. Go on to step 3.

   in the above command:
   B instructs the monitor to begin the boot sequence;
   W specifies that you wish to boot up from a Winchester disk drive;
   0 specifies Winchester disk unit 0; and
   1 specifies what you wish to boot from partition 1 on the disk.

   Notice that a colon (:) is required as a delimiter between the Winchester drive unit and partition number parameters.

NOTE: The remainder of partition startup is the same as that described in steps 5 through 8 of the sequence in Chapter 2, "Startup Procedure."

3. Wait for the MS-DOS identification message and the date prompt to be displayed.

Working Partition Procedure

4. Respond to the date prompt by entering the date in the form ***mm-dd-yy*** and pressing **RETURN**. The time prompt will be displayed.

5. Respond to the time prompt by entering the time in the form ***hh***[:*mm*[:*ss*]] and pressing **RETURN**. MS-DOS will display the system prompt (A>).

You have completed the Working Partition Procedure.

NOTE: For a complete explanation of how to boot up from a Winchester partition, refer to Chapter 6, "Bookup Features."

Part II

Primary Feature Guide

Primary Feature Guide

This guide provides a comprehensive description of the features of MS-DOS version 2 that will be beneficial to the average user.

A *feature* is any characteristic, aspect, or property of the operating system that is not exclusively related to a specific command.

Knowledge of these features is not essential for most users who only wish to run an application program under MS-DOS. However, knowledge of these features can make MS-DOS version 2 a more powerful, convenient, and versatile software tool.

These features are explained in terms that can be understood by users who have had no microcomputer experience prior to reading this manual. The only prerequisite for making full use of the Primary Feature Guide is that you understand all of the concepts and have performed all of the necessary procedures in Part I, "Preparation Guide" of this manual.

The Primary Feature Guide consists of the following five chapters:

Chapter 5, "Command Features"—This chapter explains different ways of entering commands, batch command facilities, methods of ending command execution, use of MS-DOS with only a single floppy disk drive, and recovery from errors that occur during system use.

Chapter 6, "Bootup Features"—This chapter explains different ways of starting up MS-DOS, analyzes the processes involved in starting up MS-DOS, and helps you recover from startup errors.

Chapter 7, "Directory Features"—This chapter explains how you can fully utilize the powerful hierarchical tree-structured MS-DOS disk directory to organize your software and data.

Chapter 8, "Input/Output Features"—This chapter explains how MS-DOS transfers information between hardware devices. It also explains how you can automatically send the product of an MS-DOS operation to a file, a hardware device, or a different operation.

Chapter 9, "System Component Features"—This chapter explains some of the basic characteristics and functions of the software components within MS-DOS. It also explains how you can customize MS-DOS to your personal tastes or to use an uncommon hardware device without having to reprogram the system.

Chapter 5
Command Features

Overview

A command is a program that can help you to create, change, analyze, or move data and software.

The command features of MS-DOS enable you to enter commands quickly and accurately. You may even choose automatic command execution. These features also allow you to enter commands requiring two disks—even if you have only one disk drive.

This chapter explains the different methods you can employ in entering commands to MS-DOS. This chapter also explains how you can deal with the problems that can occur when MS-DOS does not respond to your commands exactly as you had expected.

Included in this chapter are tables listing the keyboard entries for special functions, a tutorial demonstrating these special functions, a tutorial demonstrating the system's automatic command entry capabilities, and explanations of every system error message that MS-DOS displays.

Command Types

You can communicate commands to the system by entering (typing) them at the microcomputer's keyboard. You enter commands after starting up MS-DOS, when the flashing cursor is positioned to the right of the system prompt.

Command Types

MS-DOS commands belong to either of two categories: resident or transient.

Resident commands are located in the operating system itself. Therefore, resident commands are available for use almost any time MS-DOS has been started or is loaded into microcomputer memory.

Transient commands are located in an executable file that is recorded on a disk. Therefore, transient commands are available for use only when a disk containing the transient command file is in a disk drive connected to the microcomputer.

Knowing the location (resident or transient) of a command becomes more important as you use your system more extensively and as you use various application programs. You may wish to copy some transient command files to application program disks to make using the program more efficient. For example, you may wish to have the MS-DOS transient command files for formatting, copying, and checking disks on your application program disks so that you may easily prepare data and create backup disks.

Resident Commands

Resident commands are commands that are located in the COMMAND.COM component of MS-DOS. COMMAND.COM is a file that is automatically loaded into microcomputer memory when you start MS-DOS. Resident commands are available for use whenever COMMAND.COM is loaded into microcomputer memory.

MS-DOS resident commands are listed below. For more information on specific commands, refer to Chapter 11, "Command Descriptions."

BREAK
CHDIR (CD)
CLS
COPY
CTTY
DATE
DEL (ERASE)
DIR
ECHO
EXIT
FOR
GOTO
IF
MKDIR (MD)
PATH
PAUSE
PROMPT
REM
REN (RENAME)
RMDIR (RD)
SET
SHIFT
TIME
TYPE
VER
VERIFY
VOL
d:

NOTE: In this list, *d*: represents a disk drive name. When entered alone at the system prompt, this type of resident command changes the default drive to the specified drive.

You cannot see resident commands when you examine a disk's directory (with the DIR command) because they are not stored by their command names in individual files, but are part of COMMAND.COM.

COMMAND.COM provides the MS-DOS system prompt in the form of a drive name letter and a "greater than" symbol (>). For example:

```
A>_
```

The cursor is the focus of any editing actions that you perform. The cursor is indicated by an underline in this manual; though the cursor may be an underline or a block on the video screen.

At system startup, the system prompt is `A>`, assuming you booted from the upper (or only) floppy disk drive. After startup, you may change the default (that is, the currently selected drive).

To select a new drive, simply enter the designated letter followed by a colon:

```
A>_                    (prompt; default drive)
A>B: RETURN            (user enters new drive name)
B>_                    (new prompt; new default drive)
```

There are occasions when the portion of COMMAND.COM that contains the resident commands can be erased from memory. On such occasions, the system might display a message instructing you to insert a disk containing COMMAND.COM. Refer to the explanation of the transient portion of COMMAND.COM in Chapter 9, "System Component Features," for more information.

Transient Commands

Transient commands are commands that are located in executable files that are recorded on a disk.

Transient commands are available for use only when a disk containing the applicable transient command file is in a disk drive connected to the microcomputer. A *transient command file* is a file with the extension .COM or .EXE or .BAT that causes command execution.

Some of the transient commands that are provided with an operating system are often referred to as *utilities*.

A list of some of the transient commands (or utilities) provided with your MS-DOS software follows. For more information on specific commands, refer to Chapters 11 through 18.

ASSIGN
BACKUP
CHKDSK
COMMAND
COMP
CONFIGUR
DEBUG
DETECT
LIB
LINK
MAP
MODE
MORE
PART
PREP
PRINT

DISKCOMP
DISKCOPY
EDLIN
FC
FIND
FORMAT
RDCPM
RECOVER
RESTORE
SHIP
SORT
SYS
TREE

Application programs and programs created with most high level languages may be executed as transient commands, if they have extensions of .EXE or .COM.

Another kind of transient command, not listed here, is the batch file. Batch files are described later in this chapter.

Command Line Entry

All MS-DOS commands are entered at the system prompt, but resident commands and transient commands have slightly different entry rules.

To enter a resident command, you type the command name when the cursor is flashing to the right of the system prompt. (Do not enter a drive name in front of a resident command name.) With some resident commands, you will have to make additional entries in the command line.

To enter a transient command, you enter the primary file name of the transient command file when the cursor is flashing to the right of the system prompt. (Do not enter the extension of the transient command file.) With some transient commands, you will have to make additional entries in the command line.

If the disk containing the command is not in the default drive, you will have to specify the name of the drive containing the transient command. If the disk containing the command is not in either the default drive or a specified drive, MS-DOS will not be able to find and execute the command.

You can view the names of any transient commands on your disks by using the DIR resident command. The file names of transient command files can end with the extension .COM or .EXE or .BAT. MS-DOS will process a transient command differently, depending on its extension. The processing of commands is explained later in this chapter.

Parameters of a Command Line

A command line is the form of response you make to the system prompt to bring up, or invoke, a command. A command line usually consists of three parameters: the *function*, the *argument*, and the *RETURN*.

Although there are many different entry forms for MS-DOS commands, most of these entry forms consist of these three basic parameters.

All of these parameters are entered in *command lines*. Command lines must be entered in response to an MS-DOS system prompt.

The *default system prompt* consists of the letter for the default drive and the "greater than" (>) symbol. (The *default* prompt may be changed by using the PROMPT command. Refer to the section on the PROMPT command in Chapter 11, "Command Descriptions," for more information.) The cursor should flash directly to the right of the system prompt. A prompt in this form should be displayed when you start up MS-DOS, as shown.

```
A>_
```

The system prompt and cursor indicate that MS-DOS is ready to receive a command line.

Most command lines should be entered in the following general form:

```
[d:]function[argument]RETURN
```

where ***d***: is the name of the drive containing a transient command file (if located on other than default drive);

function is the name of a resident command, transient command, or a batch file;

argument is an entry (which may include more than one parameter) that regulates the way in which the command operates; and

RETURN is the entry that terminates the command line.

Alphabetic characters in a command line can be entered in uppercase, lowercase, or a combination of both. MS-DOS will automatically convert any lowercase character into the corresponding uppercase character.

Command line parameters are often separated by a *delimiter*, or separation character. MS-DOS allows you to use a space, a tab, a comma (,), a semicolon (;), an equal sign (=), or a plus sign (+) as delimiters. (Most command line examples in this manual use a space or a comma as the delimiter.)

The major parameters of a command line are similar to the major components of an English sentence.

The function of a command line is like the predicate (verb) of a sentence, because they both indicate the action being performed.

The argument of a command line is like the direct object of a sentence, because they both indicate the item(s) being acted upon and/or the way in which the action is being performed.

The RETURN entry of a command line is like the period at the end of a sentence, because they both indicate that the intentions of the speaker (or user) have been fully expressed, and that the action should begin.

The system prompt of a command line is like the subject of a sentence, because they both indicate what or who is being addressed. In a command, you should assume that you are addressing MS-DOS, which displays its system prompt. This assumption is similar to the assumption made in a sentence where the subject "you" is assumed but not written. Each command line parameter is explained in detail below.

Function

The function is the first parameter entered in a command line. The function indicates the action that is to be performed. The function can be a resident command, a transient command, or a batch file. When the function is a transient command, you may have to precede it with a drive name.

Resident Command Function

If a function is a resident command, then the command can be entered at any system prompt. Table 5.1 shows examples of valid resident command functions.

NOTE: If the purpose of a command is simply to change the logged drive, then this command should contain only the function and RETURN entry parameters, as with the ***d: RETURN*** command.

Transient Command Function

If a function is a transient command, you must be certain that the function also indicates the location of the transient command. You can specify the location of the transient command by enter-

Table 5.1. Resident Command Function Examples

FUNCTION	PURPOSE
DIR	to view contents of a directory
PROMPT	to set the system prompt
d:	to change the logged disk drive
TIME	to view and/or change the time setting

Table 5.2. Transient Command Function Examples

FUNCTION	PURPOSE
CONFIGUR	to adjust system for hardware devices, using the CONFIGUR.COM command from the default drive
d:DISKCOPY	to copy the contents of one disk to another, using the DISKCOPY.COM command from drive *d*
BATCH	to cause automatic execution of the commands listed within the BATCH.BAT batch file, which is stored in the root directory of the default drive. (BATCH is just an example, you may name batch files with any valid eight-character name, as long as .BAT is the extension used.)

ing a drive name immediately before the file name of the transient command, as shown in the following form:

d*: *command

Table 5.2 shows examples of valid transient command functions.

NOTE: Although batch files are a type of transient command function, their use is explained in detail later in this chapter.

Parameters

Parameters are entered one space after the function. The parameters indicate what data (for example, directories, files, systems, disks, drives) the function's activity involves.

Parameters can be included in your MS-DOS commands to specify additional information to the system. If you do not include some parameters, MS-DOS provides a default value. Refer to Chapter 11, "Command Descriptions," for the default values for specific commands.

Command Line Entry

The following is the general format of all MS-DOS commands:

```
function[parameters...]RETURN
```

where [***parameters***...] may consist of:

d: is the name of the designated drive.

filename is any valid (primaryfile) name for a disk file, most often including a file extension.

NOTE: The file name parameter does not refer to a device or to a disk drive name.

extension is a period and 1-3 characters immediately following a file name.

filespec is a parameter that can consist of a drive name, a path name, and/or a file name. A file specification (*filespec*) can have any of the following forms:

[*d*]:*filename*
[*d*]:*primname*[*.ext*]
[*d*]:*pathname*

pathname is the name of a path to a specified directory or file of the form:

```
[\][directory][\directory...][\filename]
```

switches are parameters that control MS-DOS commands. They are preceded by a forward slash (for example, /P).

The RETURN Entry

After entering the function and argument, you must press RETURN to tell MS-DOS that the entire command line has been entered and is ready for execution.

Command Line Requirements

MS-DOS requires that command lines be entered in a specific form. Besides the requirements that you must begin by entering a valid command name and end by pressing RETURN, there are other specific requirements that must be met.

You must always separate the command line function and the command line argument with one space. Furthermore, any command entered at the system prompt must end with a RETURN. However, commands themselves often display prompts in the course of their operation. When such a prompt (one given during the operation of a command) is displayed, a RETURN may not be required in your response.

The following requirements apply to entering all MS-DOS commands, both resident and transient:

- Command functions are usually followed by one or more parameters.
- Commands and parameters may be entered in uppercase, lowercase, or a combination of both. The system converts all alphabetic characters to uppercase.
- Commands and parameters must be separated by delimiters. Valid delimiters under MS-DOS are the space, tab, comma (,), semicolon (;), equal sign (=), and plus sign (+).
- Delimiters in addition to the colon and period must not be used within a file specification (that is, enter the file specification using the form, *d*:filename.ext). The colon and period are the only delimiters needed within any file specification.
- Sometimes the system will prompt you to perform certain steps during command execution. When the system prompts you to `Press any key`, you can press any *single* alphabetic (A-Z) or numeric (0-9) key on the keyboard.

Command Features

Command Line Entry

- You must include the file name extension as part of the parameter in a command line when referring to a file that has an extension.

- For any command line you enter, the system will not begin command execution until you press RETURN.

- If you want to abort or terminate execution of a command that is already running, you can press CTRL-BREAK (or CTRL-C). This is done by holding down the CTRL key and pressing the BREAK key simultaneously.

- Wildcards (also referred to as "global file name characters") and device names, such as CON or PRN, may not be used in any command name.

- When command execution produces a large amount of output on the terminal screen, the display of that output automatically scrolls upward to display all output. You may suspend the display (to make it easier to read) by pressing CTRL-NUMLCK. Press any key to resume the scrolling display.

- MS-DOS editing and function keys may be used when entering command lines.

- Disk drives (and sometimes files) are referred to as source drives (files) and destination drives (files). A source drive (or file) is the one *from* which you will be transferring information. A destination drive (or file) is the one *to* which you will be transferring information.

- Do not separate the parameters of a file specification with delimiters, since the colon and the period already serve as delimiters.

- Command interpretation begins when COMMAND.COM scans the command line it receives for the name of a legal command. If a legal resident command is found, the command is executed immediately. If a legal batch file name is found, commands are retrieved from a .BAT file for indirect execution. If a valid transient command (extension .EXE or .COM) is found, the appropriate file is loaded from disk into memory and then the command is executed. (Refer to the section on Batch Processing found later in this chapter.)

- If more than one executable file with the same file name exists on a disk, COMMAND.COM interprets the invoked command on the basis of the extensions the file names have. The highest priority (first-executed) files have the extension .COM; .EXE files are second-priority, and .BAT are last. For example, if you specify that you want to execute "TEST" and both TEST.COM and TEST.EXE are on the disk, COMMAND.COM will execute TEST.COM. Similarly, if both TEST.EXE and TEST.BAT are on the disk, COMMAND.COM will execute TEST.EXE.

Command Line Editing

MS-DOS is selective when it comes to accepting command lines. You must spell all parameters of a command line correctly and include the names of nondefault disks whenever a referenced file is not on the default disk. If you do not, MS-DOS is not able to execute your command. MS-DOS responds to an error such as this with the message, `Bad command or filename`.

MS-DOS does aid you in its inflexibility at interpreting a command by providing a way to edit your command entries.

This section explains the single keys and combinations of keys that you can press to edit a command line into shape before submitting it to MS-DOS for execution.

NOTE: In this text, "CTRL" followed by a hyphen and a character indicates that you should hold down the key marked CTRL (control key) and press the key marked with the specified character.

The special editing and control character functions make MS-DOS very easy to use. All of the MS-DOS commands, from

DEBUG to FC, can make use of these functions wherever input is required from a terminal. These functions are always resident in the operating system.

Command Line Template

MS-DOS offers a variety of functions that operate on the command line buffer. A *buffer* is an area set aside in memory for a specific function and usually of a specific size. The command line buffer will hold up to, and including, 128 characters. The command line buffer is intimately related to another buffer called the template. The template holds the previous contents of the command line. The template is used in many of MS-DOS's special editing functions.

The *command line buffer* is an area of memory in which the characters of a command line are stored between the time you enter them and press RETURN (command execution). Command line characters are stored in this buffer when a command is entered while the previous command is executing.

The *template* is an area of memory that contains an image of the last command line entered since bootup. You can retrieve some or all characters of the previous command from the template as you enter your next command.

When you enter text from the keyboard, it is held in the command line buffer until you press the RETURN key. At the same time, the previous input is copied to the template for use by the command line editing functions.

When you press RETURN, the contents of the command line buffer are sent to COMMAND.COM for processing.

Thus, the template contains the last command line entered to the command line buffer.

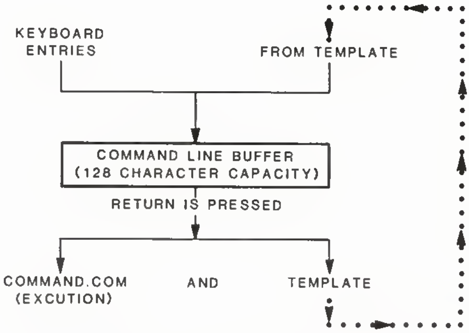


Figure 5.1. Command Line and Template

The relationship between the command line and the template is shown in Figure 5.1.

In Figure 5.1, the solid lines denote the paths that input data follows from the keyboard on the way to execution. The dotted lines denote the optional path that input data would take if the template is used to edit the command line.

As seen in Figure 5.1, you type a command to MS-DOS on the command line. When you press the RETURN key, the command is automatically sent to the command processor (COMMAND.COM) for processing. When the command line buffer is used again the previous command line is sent to the template. You can now recall the command or modify it with the MS-DOS special editing keys.

Command Line Editing Keys

The special editing keys deserve particular emphasis because they depart from the way in which most operating systems handle command input. You do not have to type the same sequences of keys repeatedly, because the last command line is automatically placed in the template.

By using the template and the special editing keys, you can take advantage of the following MS-DOS features:

1. A command line can be instantly repeated by pressing two keys.

2. If you make a mistake in the command line, you can edit it and retry without having to retype the entire command line.

3. A command line that is similar to a preceding command line can be edited and executed with a minimum of typing by pressing special editing keys.

Table 5-3 contains a complete list of the special editing keys and their functions. Each of these keys is more fully described in Chapter 12, "EDLIN."

Table 5.3. Special Editing Functions (Intraline)

KEY NAME	FUNCTION NAME	DESCRIPTION
F1	COPY1	Copies one character from the template.
F2*x*	COPYUP	Copies multiple characters from the template, up to the specified character, *x*.
F3	COPYALL	Copies all characters in the template.
DEL	SKIP1	Skips (deletes) one character in the template.
F4*x*	SKIPUP	Skips (deletes) multiple characters in the template, up to the specified character, *x*.
ESC	QUIT INPUT OR VOID	Voids the current input without affecting the template.
INS	INSERT	Invokes the insert mode, such that new input does not overwrite any characters in the template.
F5	NEW TEMPLATE	Creates new template.
F6	CTRL-Z	Puts a CTRL-Z (1AH) end-of-file character in the new template.

NOTE: The INS (insert) key is a toggle. The first time the key is pressed, the insert mode is turned on; the next time it is pressed, the insert mode is turned off.

Command Line Entry Tutorial

The following examples illustrate the operation of command line editing using the template.

Assume you have a program file called PROG.COM. If you type the following command

```
DIR PROG.COM
```

and press **RETURN**, MS-DOS displays information about the file PROG.COM on your screen. The command line is also saved in the template. To repeat the command, just press two keys: **F3** and **RETURN**.

The repeated command is displayed on the screen as you type, as shown below:

```
F3 DIR PROG.COM RETURN
```

Notice that pressing the F3 key causes the contents of the template to be copied to the command line; pressing RETURN causes the command line to be sent to the command processor for execution.

If you want to display information about a file named PROG.ASM, you can use the contents of the template and type:

```
F2 C
```

Typing F2 C copies all characters from the template to the command line, up to but not including C. MS-DOS displays:

```
DIR PROG._
```

Note that the cursor appears after the period. Now type

```
ASM
```

the result is:

```
DIR PROG.ASM_
```

The command statement `DIR PROG.ASM` is now in the template and ready to be sent to the command processor for execution. To do this, press **RETURN**.

Now assume that you want to execute the command, TYPE PROG.ASM. To do this, you would type:

```
TYPE INS F3RETURN
```

Notice that when you are typing, the characters are entered directly into the command line and overwrite corresponding characters in the template. This automatic replacement feature is turned off when you press INS. Thus, the characters *TYPE* replace the characters *DIR* in the template. To insert a space between TYPE and PROG.ASM, you press INS and then the SPACE BAR. Finally, to copy the rest of the template to the command line, you press **F3** and then **RETURN**. The command *TYPE PROG.ASM* is processed by MS-DOS, and the template becomes *TYPE PROG.ASM*.

If you had misspelled *TYPE* as *BYTE*, a command error would have occurred. Still, instead of discarding the whole command, you could save the misspelled line before you press RETURN by creating a new template with the F5 key:

```
BYTE PROG.ASM F5
```

You could then edit this erroneous command by typing:

```
T F1 P F3
```

The F1 key copies a single character from the template to the command line. The resulting command line is then the command that you want:

```
TYPE PROG.ASM
```

As an alternative, you can use the same template containing BYTE PROG.ASM and then use the F1 and INS keys to achieve the same result:

```
DEL DEL F1 INS YP F3
```

Command Line Editing

To illustrate how the command line is affected as you type, examine the keys pressed on the left; their effect on the command line is shown on the right:

KEYS	ACTION	RESULTS
DEL	Skips over 1st template character	
DEL	Skips over 2nd template character	
F1	Copies 3rd template character	T
INS YP	Inserts two characters	TYP
F3	Copies rest of template	TYPE PROG.ASM

Notice that DEL does not affect the command line. It affects the template by deleting the next character in the buffer.

Similarly, F4 deletes characters in the template, up to but not including a given character.

These special editing keys can add to your effectiveness at the keyboard. The next section describes control character functions that can also help when you are typing commands.

Control Entries

When using MS-DOS, you can make a number of control character entries. *Control character entries* alter a command line or affect the way your peripheral devices respond to a command line.

When you press a control character, such as CTRL-BREAK, you must hold down CTRL and then press BREAK.

Table 5.4. Control Entries

CONTROL CHARACTER	FUNCTION
ALT-BREAK	Flushes all data from the keyboard buffer, and returns you to the system prompt.
CTRL-BREAK	Aborts current command immediately.
CTRL-C	Aborts current command as soon as any preceding entries in the command line buffer are executed.
CTRL-H	Removes last character from command line and erases character from terminal screen.
CTRL-RETURN	Inserts RETURN but does not empty command line. Extends the current logical line beyond the physical limits of one terminal screen.
CTRL-P	Cancels echoing of output to line printer.
CTRL-NUMLCK	Suspends output display on terminal screen. Press any key to resume.
CTRL-PRTSC	Sends terminal output continuously to line printer.
SHIFT-PRTSC	Outputs to the printer everything on your screen at the time SHIFT-PRTSC is entered.

Automatic Command Entry

The MS-DOS automatic command entry feature allows you to create batch files, which are files that execute a number of functions (commands) with a minimum of keyboard entry.

With the batch file processing feature, you can create a complete sequence of commands to execute many procedures. All of which can be executed with a single command.

This feature of MS-DOS is also available via a command file that automatically executes at bootup.

The AUTOEXEC.BAT file—which automatically executes a batch file (of your creation) at bootup—may only be executed by the command processor.

The AUTOEXEC.BAT File

When you boot up your system, COMMAND.COM searches for the file AUTOEXEC.BAT. If a file with that name exists on disk, then the batch facility is automatically invoked to execute the commands contained in AUTOEXEC.BAT. This file must be placed in the root (\) directory for MS-DOS to locate and execute it. If the file exists, COMMAND.COM triggers sequential execution of the commands within the batch file. In such a case, execution of the TIME and DATE commands at startup is bypassed. If COMMAND.COM does not find AUTOEXEC.BAT, then the DATE and TIME commands are executed and the normal MS-DOS prompt is displayed instead.

An AUTOEXEC.BAT file allows you to automatically execute commands or programs when you start MS-DOS. Automatic command execution is useful when you want to run one or more specific applications automatically, each time you start the system.

For more information about the activities surrounding the execution of the AUTOEXEC.BAT file at bootup, refer to Chapter 9, "System Component Features."

Figure 5.2 illustrates how MS-DOS uses the AUTOEXEC.BAT file.

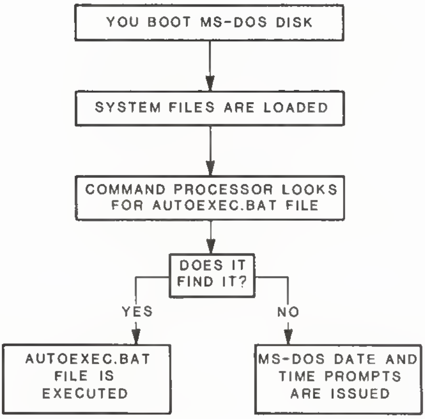

YOU BOOT MS-DOS DISK
SYSTEM FILES ARE LOADED
COMMAND PROCESSOR LOOKS FOR AUTOEXEC.BAT FILE
DOES IT FIND IT?
YES
NO
AUTOEXEC.BAT FILE IS EXECUTED
MS-DOS DATE AND TIME PROMPTS ARE ISSUED

Figure 5.2. How MS-DOS Uses the AUTOEXEC.BAT File

Creating an AUTOEXEC.BAT File

If, for example, you wanted to automatically load BASIC and run a program called MENU each time you started MS-DOS, you could create an AUTOEXEC.BAT file as follows:

1. Type

   ```
   COPY CON AUTOEXEC.BAT
   ```

   and press **RETURN**. This statement tells MS-DOS to copy the information from the console (keyboard) into the AUTOEXEC.BAT file. Note that the AUTOEXEC.BAT file must be created in the root directory of your MS-DOS disk.

2. Now type

   ```
   BASIC MENU
   ```

   and press **RETURN**. This statement goes into the AUTOEXEC.BAT file. It tells MS-DOS to load BASIC and run the MENU program whenever MS-DOS is started.

3. Press ***F6***. Then press **RETURN** to close the AUTOEXEC.BAT file.

4. The MENU program will now run automatically whenever you start MS-DOS.

To run your own BASIC program, enter the name of your program in place of MENU in the previous example. You can enter any MS-DOS command or series of commands into the AUTOEXEC.BAT file.

NOTE: Remember that if you use an AUTOEXEC.BAT file, MS-DOS will not prompt you for a current date and time unless you include the DATE and TIME commands in the AUTOEXEC.BAT file. It is strongly recommended that you include these two commands in your AUTOEXEC.BAT file, since MS-DOS uses this information to keep your directory current.

Batch Processing

Often you may find yourself typing the same sequence of commands over and over to perform some commonly used task. With MS-DOS, you can put the command sequence into a special file called a batch file, and execute the entire sequence simply by typing the name of the batch file. Batches of your commands in such files are processed as if they were typed at a terminal. Each batch file must be named with the .BAT extension. To execute the batch file, just type the file name without its extension.

You can create a batch file by using EDLIN, another text editor/ word processor, or by typing the COPY command.

Batch commands are entered at the system prompt in the following form:

```
filespec[parameter1[parameter2...[parameter9]]]
```

where *filespec* is the primary file name of a batch file, which must have a .BAT extension. Drive name and path name specifications are optional;

parameter1 is a string of characters that you wish to have automatically substituted into a location within the batch file named *filespec*. This string will be substituted into the batch file location marked by %1;

parameter2 is the character string that will be substituted into the batch file location marked by the characters %2; and

parameter9 is the character string that will be substituted into the batch file location marked by the characters %9.

Batch-Processing, Resident Commands

Although all MS-DOS commands are available for use in batch files, some are especially useful for batch processing. These commands are:

ECHO	The ECHO command is used to control the display of commands and remarks (or messages) during execution of a batch file.
FOR	The FOR command is used to repeat an MS-DOS command during batch file and interactive file processing.
GOTO	The GOTO command is used to branch unconditionally out of the normal execution sequence in a batch file to a special label in the batch file.

IF	The IF command is used to allow conditional execution of MS-DOS commands under batch processing.
PAUSE	The PAUSE command is used to suspend execution of a batch file until any entry except CTRL-BREAK is made. The PAUSE command also displays optional comments, after which it displays the message, `Strike any key when ready.` If CTRL-BREAK is entered while PAUSE is in effect, batch file execution is aborted.
REM	The REM command is used to display user-oriented comments or remarks during batch file execution, without these remarks being executed as commands. The period (.) may be used in lieu of REM.
SHIFT	The SHIFT command allows you to use replaceable parameters past the normal limit of 10.

NOTE: These commands are described in detail in Chapter 11, "Command Descriptions."

Batch-Processing Tutorial

Batch processing is useful if you want to execute several MS-DOS commands with one batch command, such as when you format and check a new disk. For example, a batch file for this purpose might look like this:

```
REM    This is a file to check new disks
REM    It is named NEWDISK.BAT
PAUSE    Insert new disk in drive B:
FORMAT B:
DIR B:
CHKDSK B:
```

To execute this .BAT file, simply type the file name without the .BAT extension, as shown

```
NEWDISK
```

and press **RETURN**. The result is the same as if each of the lines in the .BAT file was entered at the terminal as individual commands.

The following list contains rules and information that you will need to know before you execute a batch process with MS-DOS.

1. Only the primary name of the batch file should be entered to execute the batch file. Do not enter the file name extension.

2. The commands in the file named *primname*.BAT are executed.

3. If you enter CTRL-BREAK while in the batch mode, this prompt appears:

   ```
   Terminate batch job (Y/N)?
   ```

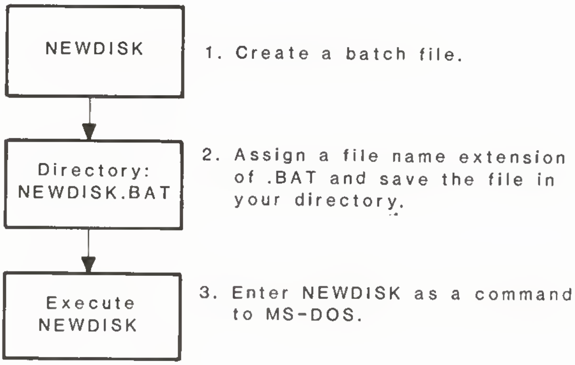
   If you press **Y**, the remainder of the commands in the batch file are ignored and the system prompt appears.

   If you press **N**, only the current command ends and batch processing continues with the next command in the file.

4. If you remove the disk containing a batch file being executed, MS-DOS prompts you to insert it again before the next command can be read.

   ```
   Insert disk with batch file
   and press any key when ready.
   ```

5. The last command in a batch file may be the name of another batch file. This allows you to call one batch file from another when the first is finished.



1. Create a batch file.
2. Assign a file name extension of .BAT and save the file in your directory.
3. Enter NEWDISK as a command to MS-DOS.

Figure 5.3. MS-DOS Batch File Steps

Figure 5.3 illustrates the three steps used to write, save, and execute an MS-DOS batch file.

Using Replaceable Parameters

Batch processing under MS-DOS allows you to specify replaceable parameters. These replaceable parameters (%0 through %9) are used within a batch file as "dummy" parameters which will be replaced sequentially with real values when the batch file is executed.

The parameters are substituted in order on the command line. A one-to-one correspondence is set up between the prototype commands in the command line and the replaceable parameters in the batch file. That is, '%1' stands for the first value (name, number, or text) typed after the batch file name, and '%2' stands for the second value typed after the batch file name. This relationship is illustrated below.

The example command line shown below corresponds on a one-to-one basis with the replaceable parameters directly under it:

filename	*filespec1*	*filespec2*	*filespec3 ...*	*filespec9*
%0	%1	%2	%3 ...	%9

NOTE: The replaceable parameter '%0' is always replaced by the drive name (if specified) and the file name of the batch file. This allows for the creation of batch file commands that can be used on more than just one set of files or can be used to *restart* themselves.

Although you may only use up to ten replaceable parameters within a batch file (%0 through %9), you can avoid this limitation by using the SHIFT command. The SHIFT command shifts the parameters to the left, one parameter at a time. The SHIFT command is described in detail in Chapter 11, "Command Descriptions."

By creating a .BAT file with "dummy" parameters, you can pass parameters to the .BAT file when it is executed. The percent sign (%) used in front of the parameter number tells COMMAND.COM that the number is a parameter that will be replaced by a real value. If you use the percent sign as part of a file name within a batch file, you must type it twice. For example, to specify the file ABC%.EXE, you must type it as **ABC%%.EXE** within the batch file.

Creating a .BAT File with Replaceable Parameters

There may be times when you want to create an application program and run it with different sets of data. These data may be stored in various MS-DOS files.

Automatic Command Entry

When used in MS-DOS commands, a parameter is an option that you define. With MS-DOS, you can create a batch (.BAT) file with dummy (replaceable) parameters. These parameters, named %0-%9, can be replaced by values supplied when the batch file executes.

For example, you can create a batch file by entering the following lines. Please note that you must press the RETURN key after each line to enter it into the system. When you type the command line `COPY CON MYFILE.BAT`, the next lines you type are copied from the console to a file named MYFILE.BAT on the default drive:

```
A>COPY CON MYFILE.BAT
COPY %1.MAC %2.MAC
TYPE %2.PRN
TYPE %0.BAT
```

After you have entered the above lines, you must press **F6**. This key enters the CTRL-Z sequence which designates the end of the batch file. Without this, the file is not legally closed. Also, you must press **RETURN** to enter the complete sequence into the system. After you have completed the entry, MS-DOS responds with this message:

```
1 File(s) copied
A>
```

The file MYFILE.BAT, which consists of three commands, now resides on the disk in the default drive.

Remember that the dummy parameters %1 and %2 are replaced sequentially by the parameters you supply when you execute the file. The dummy parameter %0 is always replaced by the drive name, if specified, and the file name of the batch file (in this case, MYFILE).

Executing a .BAT File

To execute the batch file MYFILE.BAT and to specify the parameters that will replace the dummy parameters, you must enter the batch file name (without its extension) followed by the parameters you want MS-DOS to substitute for %1, %2, and so on. Remember that the file MYFILE.BAT consists of 3 lines:

```
COPY %1.MAC %2.MAC
TYPE %2.PRN
TYPE %0.BAT
```

To execute the MYFILE batch process, type

```
MYFILE A:PROG1 B:PROG2
```

and press **RETURN**, where *PROG1* and *PROG2* are any two program files. MYFILE is substituted for %0, A:*PROG1* for %1, and B:*PROG2* for %2.

The result is the same as if you had typed each of the commands in MYFILE at the keyboard, with their parameters, as follows:

```
COPY A:PROG1.MAC B:PROG2.MAC
TYPE B:PROG2.PRN
TYPE MYFILE.BAT
```

Figure 5.4 illustrates how MS-DOS replaces each of the above parameters.

BATCH FILE NAME	PARAMETER 1 (%0) (MYFILE)	PARAMETER 2 (%1) (*PROG1*)	PARAMETER 3 (%2) (*PROG2*)
MYFILE	MYFILE.BAT	*PROG1*.MAC	*PROG2*.MAC *PROG2*.PRN

Figure 5.4. Parameter Values for MYFILE.BAT

Remember that the dummy parameter %0 is always replaced by the drive name (if specified) and the file name of the batch file.

Example Batch Files

The following examples illustrate the creation of batch files and the different methods used to create them.

Example 1, Creating a Batch File with EDLIN

The EDLIN text editor can be used to create batch files, instead of creating them directly from console input. To do this, type

```
A:EDLIN NEWDISK.BAT
```

and press **RETURN**. You must give the batch file you create a .BAT extension unless you want to use REN to later rename the file with .BAT as its extension.

Now, enter the following batch file by using EDLIN.

NOTE: A RETURN is required after all lines (except 7) below.

```
*I

     1:* REM This is file NEWDISK.BAT
     2:* REM (the .BAT extension must be given)
     3:* PAUSE Insert disk in B:
     4:* FORMAT B:/S
     5:* DIR B:
     6:* CHKDSK B:
     7:* CTRL-BREAK
*E
```

NOTE: In line 7, you must hold down the CTRL key while pressing the BREAK key. (Do not type the letters C T R L - B R E A K.)

The file NEWDISK.BAT now resides on the disk in drive A.

Execute this .BAT file by entering the file name without the .BAT extension:

```
A>NEWDISK
```

The result is the same as if you entered each of the lines in the .BAT file as individual commands.

To pass parameters to the .BAT file, create a .BAT file containing prototype commands with dummy entries.

For more detailed information on creating files with EDLIN, refer to Chapter 12, "EDLIN."

Example 2, Creating a Batch File to Assemble Source Code Files

Since batch files can contain anything that can be entered at the keyboard, you can create a batch file to assemble source code files. This has been done below.

In the next example, a RETURN is required at the end of each line. No prompts will appear—continue with the next line.

Enter the following batch file creation sequence:

```
A>COPY CON ASMFILE.BAT
REM This is A:ASMFILE.BAT
REM START BATCH FILE
COPY %1.ASM %2.ASM
MASM %2,%2,%2;
TYPE %2.PRN
TYPE %0.BAT
F6
```

The file ASMFILE.BAT now resides on the disk in drive A.

To execute this .BAT file and pass parameters, enter:

```
A>ASMFILE A:MYPROG B:MYPROG
```

The result is the same as if you had entered each of the following commands at your keyboard:

```
A>REM This is A:ASMFILE.BAT
A>REM START BATCH FILE
A>COPY A:MYPROG.ASM B:MYPROG.ASM
A>MASM B:MYPROG,B:MYPROG,B:MYPROG;
A>TYPE B:MYPROG.PRN
A>TYPE A:ASMFILE.BAT
```

Example 3, Creating a Batch File to Check for File Existence

The batch file created by the following commands can be used to check that a file exists. To create this batch file, enter

```
A>COPY CON CHECK.BAT
ECHO OFF
IF EXIST %1 GOTO X
ECHO SORRY, THAT FILE HAS BEEN ERASED GOTO END
GOTO END
:X
ECHO THAT FILE IS ON DISK
:END
```

and press the **F6** key to close the file. Press the **RETURN** key to store the file. You will get the following message on your screen:

```
            1 File(s) copied
A>
```

You can now use the batch file CHECK.BAT to to see if a file exists on your disk.

Example 4, Creating a Batch File to Examine the Contents of Files

The following example uses the TYPE command within a batch file. The replaceable parameters are replaced with the file names specified on the command line. To create this batch file, enter

```
A>COPY CON EXAMINE.BAT
FOR %%A IN (%1 %2 %3) DO TYPE %%A
```

and press the **F6** key to close the file. Press the **RETURN** key to store it.

To view the contents of the files CHAP1, CHAP2, and CHAP3 in succession, type

```
EXAMINE CHAP1.TXT CHAP2.TXT CHAP3.TXT
```

and press **RETURN**.

The command says, essentially: for each parameter specified in the set, let the variable '%%A' represent one of the set (the chapters) and DO the specified command (in this case, TYPE) on '%%A'. The first time through, CHAP1.TXT becomes the value of variable '%%A' and a TYPE is executed, which displays that file. CHAP2.TXT is displayed next, and CHAP3.TXT after that.

Example 5, Creating a Batch File to Erase a Number of Files

You may wish to create a batch file like the one shown below, where the SHIFT command is used, in conjunction with the ERASE command, to delete a number of bad files.

Automatic Command Entry

In the following example, the batch file, REMOVE.BAT, is used to delete a number of bad files by using a single command:

```
A>COPY CON REMOVE.BAT
ECHO OFF
:LOOP
IF "%1"=="" GOTO DONE
ERASE %1
SHIFT
GOTO LOOP
:DONE
ECHO FILE(S) ERASED
```

Again, note that you must press **F6** and **RETURN** to close the batch file and store it in the system. This example utilizes most of the resident, batch-processing commands to create a batch file with a very useful purpose. When REMOVE.BAT is executed for a given file name, it will erase any bad files. (IF "%1"=="" the file will be erased.) If the file is not found, the screen will display:

```
File not found
FILE(S) ERASED
```

To use REMOVE.BAT to erase a number of files, type

```
REMOVE MYFILE.DOC YOURFILE.DOC HERFILE.DOC
```

and press **RETURN**. The %1 replaceable parameter will be replaced by MYFILE.DOC. After it is erased, the next item in the command line will be shifted into %1 and erased. This process continues until all items specified in the command line have been erased.

Instructions for Single-Disk Drive Users

For single-disk drive users, the commands are exactly the same syntax as for double-disk drive users. The difference lies in your perception of the "arrangement" of the drives.

Instructions for Single-Disk Drive Users

You must think of this system as having two disk drives: drive A and drive B. However, instead of A and B designating physical disk drive mechanisms, the A and B designate disks. Therefore, when you specify drive B while operating on drive A (the prompt is a:), MS-DOS prompts you to `switch drives` by swapping disks.

The prompts are:

```
Place disk A in drive B:
Press any key when ready.

Place disk B in drive A:
Press any key when ready.
```

These procedures apply to any MS-DOS COMMAND commands (both system and file) that can request or direct a different drive as part of its syntax. These commands include:

CHKDSK
COPY
DEL
DIR
DISKCOMP
DISKCOPY
ERASE
FORMAT
REN
TYPE

Also, if any of these commands are used in a batch file and call for a different drive, the single-disk drive procedures apply. Execution is halted and the appropriate prompt is displayed.

The following example may serve as an illustration for all of the commands listed above. To execute, enter

```
A>COPY COMMAND.COM B:
```

and press **RETURN**. Your screen will display the message:

```
Place disk B in drive A:
Press any key when ready.
```

You should now press any alphabetic (A-Z), or numeric (1-0) key. The screen will respond with the message:

```
1 File(s) copied
```

Command Interruption

There are two ways for the command processor to be interrupted. One is intentional, in that you initiate the interruption for a specific reason. This is known as a "control-break," because the CTRL key is held down while the BREAK key is pressed. The other way that the command processor can be interrupted is when some type of disk error has occurred; this is unintentional, in that the command processor will automatically interrupt itself.

The CTRL-BREAK function allows you to interrupt almost any activity that the microcomputer is engaged in, at any time. If you wish to stop the action of a certain command, or just get your system prompt back on the screen, you can use CTRL-BREAK. The CTRL-BREAK is similar to CTRL-C, but the effect of CTRL-BREAK is immediate, because it bypasses the keyboard buffer.

Operating System Error Messages

MS-DOS displays error messages when it encounters problems while attempting some system functions. This section explains how these error messages might occur and, when possible, how you can recover from them.

```
type ERROR WHILE i/o-action ON DRIVE d
Abort, Retry, Ignore:_
```

where *type* may be one of the following:

WRITE PROTECT
NOT READY
SEEK
DATA
SECTOR NOT FOUND
WRITE FAULT
READ FAULT
DISK

i/o-action may be either of the following:

READING
WRITING

d indicates the drive in which the error has occurred.

EXPLANATION: If a floppy disk error occurs at any time during any command or program, MS-DOS retries the operation three times. If the operation cannot be completed successfully, MS-DOS returns an error message in this form. MS-DOS waits for you to enter one of the following responses:

A	Abort.	Terminate the program requesting the disk read or write.
I	Ignore.	Ignore the bad sector and pretend the error did not occur.
R	Retry.	Repeat the operation. This response is particularly useful if the operator has corrected the error (such as with NOT READY or WRITE PROTECT).

Usually, you will want to attempt recovery by entering responses in the order:

R	(to try again)
A	(to terminate program)

Operating System Error Messages

One other error message might be related to a faulty disk's read or write operation:

```
FILE ALLOCATION TABLE BAD FOR DRIVE d
```

EXPLANATION: This message means that the copy in memory of one of the allocation tables has pointers to nonexistent blocks. Possibly the disk was not formatted before use. Reset and reboot the system. Run the CHKDSK command without specifying the /F switch. If CHKDSK finds errors, then you can run CHKDSK with the /F switch to fix disk errors.

```
type error i/o-action drive d
Sector address of error is nnnn
Abort, Retry, Ignore:
```

where *type* indicates the type of problem that caused the error condition. This problem might be worded as:

Write Protect
SEEK
DATA
SECTOR NOT FOUND
WRITE FAULT
DISK

i/o-action identifies the operation that was being performed when the error occurred. This operation could be worded as:

READING
WRITING

d is the name of the drive to which the partition was assigned when an error was encountered on the partition; and

nnnn is the logical hexadecimal address of the sector on which the hard error occurred. (Logical sector addresses begin with the first sector on the entire Winchester disk, which is sector 0000.)

EXPLANATION: When bad sectors are encountered during Winchester disk access operations, MS-DOS displays a Winchester disk error message that is slightly different from the floppy disk error message. This message appears in this form. Record the sector address of the error when this hexadecimal value is displayed in an error message. Also, record the identity of the partition on which the error(s) occurred. Press the R key to try to recover from this error condition. If this error occurs repeatedly and you cannot recover by pressing R, then run BACKUP, DETECT, FORMAT, and RESTORE in sequence.

```
Bad call format
```

EXPLANATION: The parameters that passed to a device driver are invalid. If you have installed a device driver that you created or acquired yourself (using the CONFIG.SYS file), then change this driver or install a different driver. If you have not installed a device driver through the CONFIG.SYS file, then contact your Technical Consultation for assistance.

```
Bad command or filename
```

EXPLANATION: The command you entered does not exist on the disk you are trying to access.

```
Bad or missing Command Interpreter
```

EXPLANATION: At the time the bootup command line was entered, the default command interpreter (COMMAND.COM) or the command interpreter specified in the CONFIG.SYS file (if any) was not stored in the root directory of the default or specified disk or partition. Copy COMMAND.COM to the root directory of this disk or partition, or specify a different command interpreter through the CONFIG.SYS file. Then, boot up again.

```
Bad unit
```

EXPLANATION: The parameters that passed to a device driver are invalid. If you have installed a device driver that you created or acquired yourself (using the CONFIG.SYS file), then change

Operating System Error Messages

this driver or install a different driver. If you have not installed a device driver through the CONFIG.SYS file, then contact your Technical Consultation for assistance.

```
Batch file missing

Insert disk with batch file
and press any key when ready.
```

EXPLANATION: You removed a disk containing the batch file that is currently running; in order for the batch file to finish, you must reinsert the disk.

```
Cannot do binary reads from a device
```

EXPLANATION: MS-DOS cannot read binary data from a device because data from such a source will not end with an end-of-file character (`1AH`) to help MS-DOS determine when to stop reading. If you want to read binary data from a device, you must write or acquire a program that reads binary data and determines end-of-file by a communications protocol other than an end-of-file character.

```
Content of destination lost before copy
```

EXPLANATION: It is easy to enter a concatenation COPY command where one of the source files is the same as the destination; yet this often cannot be detected. For example, the following command is an error if ALL.LST already exists:

```
A>COPY *.LST  ALL.LST
```

This is not detected, however, until it is ALL.LST's turn to be appended. At this point, it could already have been destroyed.

The COPY commands handles this problem like this: as each input file is found, its name is compared with the destination. If they are the same, that one input file is skipped and the message `Content of destination lost before copy` is printed. Further concatenation proceeds normally.

```
Duplicate filename or
File not found
```

EXPLANATION: This message can occur if you used the RENAME command to:

- replace the name of a file that does not exist on the default or specified drive or
- substitute a name for a file when the name to be substituted is already in use by a file in the default or specified drive.

```
EXEC failure
```

EXPLANATION: This message could be caused by any of the following error conditions:

- The executable file does not exist as specified. Specify the file by using the proper file name and, if necessary, the proper drive name and path name.
- The specified file is an .EXE file containing header information that is inconsistent with the characteristics of standard .EXE files.
- Your microcomputer has insufficient Random Access Memory (RAM). Acquire and install additional memory circuitry.
- You have specified so much buffer space (through the CONFIG.SYS file) that the remaining RAM is insufficient to execute the program. Specify less buffer space through the CONFIG.SYS file.
- You have loaded "Terminate But Stay Resident commands" (such as PSCMX80) so that the remaining RAM is insufficient to execute the program. Reset, reboot, and refrain from loading "Terminate But Stay Resident commands."
- You have used an invalid function. Use function number 0, 1, or 3 instead.

- You have exceeded the limited amount of space allocated for the environment (a series of ASCII strings that are used by executable programs) with ASCII values set by the SET command. Redefine these values by using the SET command to use less of the environment space.

After performing any of these solutions, try again to execute the program that you were trying to execute when the error message occurred.

```
File cannot be copied onto itself
0 File(s) copied
```

EXPLANATION: During a COPY command, if the first file specification (source) references a file that is on the default drive and the second file specification (destination) is not given, the COPY command will be aborted. (Copying a file to itself is not allowed.)

```
File creation error
```

EXPLANATION: The directory of your disk or partition is full. Eliminate some of the file entries from the disk or partition you are using, or use a disk or partition with more free directory space. (This error can occur if you have run out of directory space and still have free file space on a disk or partition.)

```
Insufficient disk space
```

EXPLANATION: The disk you are trying to save data on is full; replace with a disk that has more space.

```
Intermediate file error during pipe
```

EXPLANATION: You ran out of disk space or directory space while piping. (*Piping* is entering a command line that contains more than one command function, separated by a vertical bar. A piping operation causes two temporary files to be written on the disk.) To pipe commands without causing this error, clear extra free space on the disk or use a disk that has more free space.

```
Invalid date
Enter new date:
```

EXPLANATION: If the parameters or separators are invalid, MS-DOS returns the above message and waits for you to enter a valid date.

```
Invalid device
```

EXPLANATION: You tried to access a device that the system does not support, or that the system *does* support, but that is not connected at the present time.

```
Invalid directory
```

EXPLANATION: You tried to access or specify a directory which does not exist; double-check and reenter.

```
Invalid drive specification
```

EXPLANATION: You entered an illegal drive name; repeat the command with a legal drive specification.

```
Invalid number of parameters
```

EXPLANATION: You entered too many or too few parameters on the command line; double-check and reenter.

```
Invalid parameter
```

EXPLANATION: One of the parameters within your command line was illegal; reenter the statement with legal parameters.

```
Invalid path, not directory,
or directory not empty
```

EXPLANATION: The directory path name you entered is not a directory, but a file.

```
Invalid path or file name
```

EXPLANATION: The path name or file name you entered was invalid; double-check and reenter.

Operating System Error Messages

```
Invalid time
Enter new time:
```

EXPLANATION: If the parameters or separators are invalid, MS-DOS returns the above message and waits for you to enter a valid time.

```
Out of environment space
```

EXPLANATION: The *environment* is a series of ASCII strings that are used by executable programs. This message is displayed if you try to set a value in the system environment and there is not sufficient memory available to contain the variable name and its value. Approximately 200 bytes of memory are allocated for the system environment.

If this message is displayed, enter SET and press RETURN to display the current contents of the system environment. If there are some variable names that you no longer need, delete them and then reenter the SET command.

```
Sector not found
Abort, Ignore, Retry:
```

EXPLANATION: Press the **R** key to retry the operation. If the message appears when you retry the operation, it is likely that the disk media has become defective. If the defective media is on a floppy disk, run BACKUP, FORMAT, and RESTORE. If the defective media is on a Winchester disk partition, run BACKUP, DETECT, FORMAT, and RESTORE.

```
Syntax error
```

EXPLANATION: The syntax of your command line was illegal; and locating this command in the Index, consult the section of the manual relating to the commands in question and reenter.

```
No System
```

EXPLANATION: This error message occurs if you try to boot a disk or partition that was formatted without the /S switch (the /S switch places the system tracks on disks during the formatting process).

When the boot loader is loaded into memory during bootup, it will look at the first sectors of the data area to find and load IO.SYS and MSDOS.SYS. If these files are not found the No System error message is displayed.

```
Bad or missing Command Interpreter
```

EXPLANATION: If COMMAND.COM (or another command interpreter specified in the CONFIG.SYS file) does not reside on a disk or partition when you try to boot up this media, this error message is displayed.

The COMMAND.COM file can be recorded on any part of the data area that is available beyond the MSDOS.SYS file. COMMAND.COM is not given hidden file status. Therefore you can view it with the DIR command, copy it with the COPY command, or delete it with the DEL command.

The COMMAND.COM file is not recorded on the disk or partition by SYS. If you use SYS, you should record COMMAND.COM (or another suitable command interpreter) on the disk or partition with the COPY command. Because COMMAND.COM can be recorded anywhere on the disk or partition (beyond the sectors occupied by MSDOS.SYS), you can copy other files to the disk or partition before copying COMMAND.COM to the disk or partition.

Summary

MS-DOS uses two types of commands: resident and transient. Resident commands are loaded when you boot up and executed when you type the command function at the system prompt. Transient commands are loaded and executed when you enter the command function at the system prompt.

A command line can include the following basic command line parameters (represented by lowercase variables):

d:	The drive name.
function	The command or batch file name.
argument	The parameters (such as file specifications or switches) that are entered between the function and the RETURN to regulate the way in which the command operates.
RETURN	The command line terminator, which enters the command(s) to the system for execution.

Command lines must be entered in a specific form. Commands entered in an incorrect form will not be executed.

The command line template and command line editing keys make command line entry more convenient.

You can enter special key sequences called "control entries" at the system prompt to affect the way in which command input and output travels.

Automatic command execution involves the following activities:

- Creating and using the AUTOEXEC.BAT file.
- Creating and executing batch files to accomplish complicated tasks.
- Using replaceable parameters within batch files.

Summary

You can interrupt execution of many commands by entering CTRL-BREAK.

You can cause a microcomputer with one disk drive to do many of the things that can be done using a microcomputer with two disk drives.

Sometimes, normal use of the system will be interrupted by the display of error messages. When this occurs, respond to these messages according to the explanations provided in this chapter.

Chapter 6
Bootup Features

Overview

As you learned in Chapter 2, "Startup Procedures," *bootup* is the process that moves a copy of the operating system from the system disk to microcomputer memory, where it can function with its own commands and/or application programs.

In this chapter you will learn about the two methods through which the microcomputer can boot up MS-DOS. First, the microcomputer can be instructed to *automatically* boot the operating system when power is applied to the microcomputer. Second, the microcomputer can be instructed to wait for a *manual* Boot command to be entered before it attempts to boot the operating system.

This chapter shows you how to boot up from any 5.25-inch floppy disk drive or Winchester disk partition that is properly connected to your microcomputer. Through slight alterations of your hardware settings and/or the software on your Winchester disk, you can also determine whether you will need to type an explicitly detailed bootup command line, a short ambiguous bootup command line, or no bootup command line at all.

Manual Bootup Command Entry Form

The following entry form applies if your microcomputer is not set to automatically boot.

```
B[disk] [unit] [:partition]
```

where ***B*** is required input that informs the firmware to begin the bootup sequence;

disk is either F (floppy disk drive) or W (Winchester disk) and represents the disk to be booted, (F is the powerup default when your microcomputer is shipped);

unit is the optional unit number (0-3 for floppy disk drive units or 0-7 for Winchester disk drive units.) Unit 0 is the powerup default unit; and

:partition is a number (1, 2, 3, or 4) identifying the partition of the Winchester disk that you wish to boot. If entered, this number must be preceded by a colon (:).

Preliminary Concepts

As shipped, microcomputers with a Winchester disk drive are set for autoboot from the Winchester disk; microcomputers without a Winchester disk drive are set for autoboot from the (primary) floppy disk drive. That is, each time your microcomputer is turned on, it will boot up the operating system, automatically loading MS-DOS from disk or partition into memory. The area of memory where the operating system is loaded is referred to as *system memory*.

Software/Hardware/Firmware Interaction

Before the boot sequence is explained, it is important to understand the relationship of software, hardware, and firmware. *Software* is a general term that applies to any program (set of instructions) such as language, application, and utility programs that may be loaded into a microcomputer from any source and enable hardware operation. *Hardware* is the mechanical equipment, such as the microcomputer, used for processing data. *Firmware* is a type of software program that permanently exists in the Read Only Memory (ROM) of your microcomputer.

The *MFM-150 monitor* is the name given to the firmware resident in the microcomputer. This firmware contains the basic Input/Output (I/O) drivers used by the operating system for communicating with the machine. *Input/Output (I/O) drivers* are a set of programmed instructions that handle the movement of data between the ROM and the various devices attached through the I/O ports, such as a printer or plotter. The MFM-150 monitor firmware establishes a communications link between MS-DOS (which is software) and the various physical parts of the microcomputer (which is hardware).

The MFM-150 monitor can boot up from any drive that is properly attached to the microcomputer. The bootup sequence is so named because, by means of this sequence, MS-DOS "pulls itself up by its bootstraps." The term "bootstrap" has been shortened by common usage to boot up or boot. This sequence enables MS-DOS (with help from the MFM-150 monitor) to lift itself from the disk and into the microcomputer's memory. Once MS-DOS is installed in memory, it issues instructions and coordinates the actions of the appropriate parts of the microcomputer.

Bootup Sequence

During the automatic bootup sequence, the monitor checks either the disk in drive A (refer to your microcomputer User's Guide") or the Winchester disk for a special program that is used to boot up the operating system (MS-DOS). (During manual bootup, the monitor will check the drive you specified in the Boot Command.) This program is called the *boot loader*. Once the boot loader is found, the monitor will load and execute it. In turn, the boot loader will load the rest of the operating system from the disk. This sequence of events is known as *bootup*.

Preliminary Concepts

The boot loader is a part of MS-DOS and is stored on the Distribution Disk beginning at sector 0 of track 0. The boot loader is recorded on a disk whenever MS-DOS formats the disk. If the monitor finds the boot loader, it assumes that the disk is prepared for the operating system.

After the boot loader is found, it is copied from the disk into memory and continues the bootup sequence. The boot loader locates and copies (loads) the system from the disk into memory. (If the operating system files are not present, the boot loader informs you by displaying an error message.) Once the operating system is in memory, it examines the hardware environment to determine characteristics (such as memory size, location and type of peripherals), prepares both itself and the hardware for operation, and then takes control.

The MS-DOS Components

MS-DOS is composed of three physical components. When MS-DOS is copied from the disk into memory during the bootup procedure, the three parameters, or files, that make up MS-DOS are transferred. These files are:

IO.SYS
MSDOS.SYS
COMMAND.COM

When the boot loader is first copied into memory, the boot loader checks the disk's directory to make certain that the first file listed is IO.SYS. The boot loader then loads this file into memory. If this file is not the first file in the directory, an error message is displayed. (For more information about system component behavior during bootup as well as the MS-DOS components, refer to Chapter 9, "System Component Features.")

Bootup Methods

MS-DOS and your hardware enable you to boot up in a variety of ways. Through alterations of your hardware settings and/or the software on your Winchester disk, you can determine whether you need to type an explicitly detailed bootup command line, a short ambiguous bootup command line, or no bootup command line at all.

Automatic Bootup

The method you use to boot your microcomputer is determined by a switch setting inside the microcomputer. You do not need to change this switch unless you would prefer a different setting. The switch is numbered SW-102 (see Figure 6.1). This switch selects the default boot device and comes preset from the factory for automatic bootup. When shipped, microcomputers with a Winchester disk drive are set to boot automatically from Winchester drive unit 0, microcomputers that do not have a Winchester disk drive are set to boot automatically from floppy drive unit 0.

To determine if your microcomputer is set for automatic bootup without checking switch SW-102, power up the microcomputer when it is fully connected to its components and to an appropriate power supply. If the MS-DOS identification message or a `DISK ERROR` error message is displayed within a minute after powerup (without any keyboard entries), then the SW-102 switch is set for automatic bootup.

The default bootup device is the one that is booted when the automatic boot up takes place, or when you issue the Boot command without specifying a drive.

NOTE: If your microcomputer is set for automatic boot and you do not want to boot up to the default drive or partition, you can disable the normal bootup sequence and return to the monitor by pressing the ESC key or by pressing CTRL-BREAK. You must make the entry immediately after power up or system reset; otherwise, the system will begin automatic bootup.

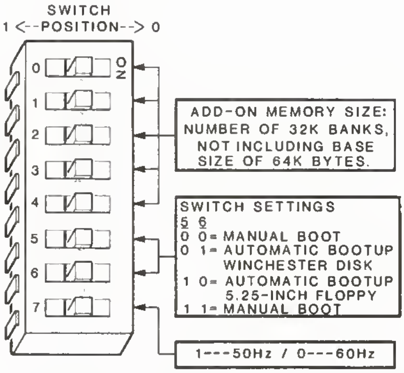

SWITCH
1 <--POSITION--> 0
ON
0
1
2
3
4
5
6
7
ADD-ON MEMORY SIZE: NUMBER OF 32K BANKS, NOT INCLUDING BASE SIZE OF 64K BYTES.
SWITCH SETTINGS
5 6
0 0= MANUAL BOOT
0 1= AUTOMATIC BOOTUP WINCHESTER DISK
1 0= AUTOMATIC BOOTUP 5.25-INCH FLOPPY
1 1= MANUAL BOOT
1---50Hz / 0---60Hz

Figure 6.1. **Hardware Bootup Settings (Switch SW-102)**

Automatic bootup from 5.25-inch floppy drive unit 0 is accomplished by placing section 5 of switch SW-102 in the 1 position and section 6 in the 0 position.

Automatic bootup from Winchester disk unit 0 is accomplished by placing section 5 of switch SW-102 in the 0 position and section 6 in the 1 position.

Manual Bootup

Once you have all your peripheral hardware devices and the microcomputer turned on, either the MS-DOS identification message or a `DISK ERROR` error message (which both indicate automatic bootup) or an MFM-150 identification message and prompt automatically appear in the upper lefthand corner of the video screen.

If an identification message and monitor prompt -> appear, similiar to the following form,

```
MFM-Monitor, Version 1.0
Memory Size: nnnK bytes
Enter "?" for help.
->
```

then manual bootup can be performed.

NOTE: If, at this point, you enter **?** and press **RETURN**, a list of commands available in the MFM-150 monitor will appear. A diagram and example of the boot command line syntax are listed. For more information on the monitor commands other than the Boot command, refer to your microcomputer User's Guide.

Manual bootup requires you to enter a Boot command. A Boot command includes the letter B and (optionally) the specification of disk, unit, and/or partition.

Explicit Bootup Commands

The MFM-150 firmware Boot command is performed from the keyboard in response to the monitor prompt. There are several options available so that you can boot from any of the drives in your system.

To begin manual bootup, enter the Boot command in the form

```
B[disk][unit][:partition]
```

where ***B*** is required input that informs the firmware to begin the bootup sequence.

disk variable represents the type of disk to be booted. The two types of disk drives are F, for floppy; or W, for Winchester. The floppy disk drive is the powerup default. (A default is the value that the system assumes if nothing is specified.)

Bootup Methods

unit variable represents the drive number to be booted from. The maximum number of floppy disk drives allowed is four. Thus, 0, 1, 2, and 3 are the possible unit specifications for floppy disk drives. The maximum number of Winchester disks allowed is eight. Thus, 0, 1, 2, 3, 4, 5, 6, and 7 are the possible unit specifications for Winchester disks. The 0 unit is the powerup default for both the floppy and Winchester disks.

:partition variable is a number, from 1 to 4, identifying the particular partition that you wish to boot. If entered, the number must be preceded by a colon (:).

Press RETURN at the end of the bootup command line.

Since the Winchester disk can contain separate partitions, unless you use the default boot partition, you must specify which partition to access whenever you enter a Boot command. Partitions are accessed by Boot commands that include or imply (in the case of the default partition) a partition.

The partition table can help you access a particular partition once you have accessed the Winchester disk itself. For further information on creating a default partition and accessing partitions, refer to Chapter 17, "PART."

Ambiguous Commands

An *ambiguous* command is the shortest entry (usually not specific) possible for a command to be carried out. For the Boot command, the results of the ambiguous entry is dependent on your hardware. The most ambiguous Boot command that can be entered is *B*. When you press **B** and **RETURN**, the MFM-150 monitor immediately attempts to boot unit 0 of the device specified by switch SW-102. (When your microcomputer is shipped, the device is a floppy disk.)

NOTE: If this is *not* the first Boot command entered since the microcomputer was turned on, refer to the section on Dynamic Bootup Parameter Defaults which follows.

If switch SW-102 is set to boot up the floppy disk, and you press **B** and **RETURN**, and you do not have a system in the floppy disk drive, the `+++ DISK ERROR: Drive not ready! +++` error message will appear. At this point, you can

- reboot by placing a system disk into the floppy disk unit 0, holding down the **CTRL** and **ALT** keys, and pressing the **DEL** key; or
- exit to the MFM-150 monitor prompt by holding down the **CTRL** key and pressing the **BREAK** key.

NOTE: Whenever a MFM-150 monitor error message other than `+++ DISK ERROR: Drive not ready! +++` is received, you must enter CTRL-BREAK or CTRL-ALT-DEL to acknowledge the error and return to the system.

When entering ambiguous commands after you initially power up your microcomputer, it is important to note the following:

- If you do not specify the disk parameter, and your microcomputer is set for automatic boot, the default will be the device specified in the SW-102 switch.
- If you do not specify the unit parameter, the default is always 0.
- If you do not specify a partition parameter, the default is either :1 or the default boot partition you specified with the PART command. (See Chapter 17, "PART.")

Dynamic Bootup Parameter Defaults

Dynamic bootup parameter defaults are bootup entry values that are stored by the MFM-150 firmware the first time you enter an explicit Boot command after powerup. These defaults enable you to enter a complete explicit Boot command (or some variation of it) several times without typing every character each time.

The MFM-150 firmware allows dynamic defaults for all parameters in the bootup command line entry form. A dynamic default is a value which, instead of remaining constant, refers back to the value entered in the most recent bootup command line.

Thus, if you enter an ambiguous command and it is *not* the first Boot command entered since the microcomputer was turned on, the MFM-150 firmware will boot the device you specify, but any omitted parameters will retain the value used in the last Boot command.

As long as the microcomputer is on, the system will retain the most recently entered Boot command. Once you turn the power to the microcomputer off, the dynamic bootup parameter defaults are erased and the defaults become the switch SW-102 setting, unit 0.

NOTE: If you enter **CTRL-ALT-DEL** when your SW-102 switch is set for automatic bootup, the system will attempt to reboot according to the switch setting and any default parameters will be erased. To return to the monitor prompt and retain the parameter defaults, you must enter **CTRL-ALT-INS**.

For example, your SW-102 switch is set for automatic bootup of the Winchester disk (unit 0, partition :1). You boot up, and then decide you want to boot up partition :3. Restart the system by entering **CTRL-ALT-INS**. When the monitor prompt appears, enter and press **RETURN**. The monitor refers to the defaults currently stored in memory. (If your microcomputer is set to automatically boot from the floppy, this default is equivalent to entering BF0. For an automatic boot from the Winchester, the default is BW0:1.) In this case, the monitor modifies the defaults to the information you have entered and fills in the missing parameters. Thus, even though you only typed B:3, the monitor behaves as though you typed BW0:3.

Now you want to boot up floppy unit 0. Enter **CTRL-ALT-INS**; enter

```
BF
```

and press **RETURN**. The MFM-150 firmware reads the BF as BF0, and since the floppy disk does not have a partition, the system temporarily ignores the dynamic default (:3) for the partition parameter. The MFM-150 firmware will store the partition default until the next time the Winchester disk is booted.

When you enter **CTRL-ALT-INS**, followed by entering

```
BW
```

and press **RETURN**, the monitor reads BW as BW0:3.

By using the dynamic bootup parameter defaults, you can take advantage of the following MS-DOS features:

- A Boot command can be instantly repeated by pressing two keys (B and RETURN).
- Variations of the Boot command can be edited and executed without having to retype the entire bootup command line.

Bootup Methods

Table 6.1. Floppy Drive Bootup Commands

FLOPPY UNIT	COMMAND ENTRY
0	B *RETURN* BF *RETURN* B0 *RETURN* BF0 *RETURN*
1	B1 *RETURN* BF1 *RETURN*
2	B2 *RETURN* BF2 *RETURN*
3	B3 *RETURN* BF3 *RETURN*

NOTE: You may get as many as two floppy units (units 0 and 1) with a microcomputer.

Example Manual Boot Commands

Table 6.1 and Table 6.2 illustrate several of the possible Boot commands you can enter in order to access a particular drive device. The last command listed for each device in this table is the most explicit command; this command will always work, regardless of the setting of the SW-102 switch.

Table 6.2. Winchester Disk Partition Bootup Commands

UNIT	PARTITION	COMMAND ENTRY
0	1	`BW` *RETURN* `BW0` *RETURN* `BW0:1` *RETURN*
	2	`BW:2` *RETURN* `BW0:2` *RETURN*
	3	`BW:3` *RETURN* `BW0:3` *RETURN*
	4	`BW:4` *RETURN* `BW0:4` *RETURN*
1	1	`BW1` *RETURN* `BW1:1` *RETURN*
	2	`BW1:2` *RETURN*
	3	`BW1:3` *RETURN*
	4	`BW1:4` *RETURN*

NOTE: You may receive only one Winchester unit (unit 0) with a microcomputer. To save space, Table 6.2 shows the command for units 0 and 1, but not for units 2–7.

To avoid obtaining Winchester disk partition error messages, you can do one or both of the following:

- Carefully type an explicit Boot command whenever you boot up to the Winchester disk. This explicit Boot command should include the number of an established partition.
- Use the PART utility to select an established partition as your "default boot partition," so that this partition will be accessed when you enter Boot commands to the Winchester disk without specifying a partition.

Bootup Results

Once the Boot command has been given, MS-DOS loads the system. When the system is loaded and you have successfully completed a bootup sequence, MS-DOS can respond in a variety of ways.

If an identification message and date prompt appear in the following form,

```
MS-DOS version 2.00
Copyright 1981,82,83 Microsoft Corp.

Current date is day mm-dd-19yy
Enter new date:
```

type the current date and press **RETURN**. MS-DOS responds with the time prompt. Type the current time and press **RETURN**. (For valid entries to the date and time prompts, see Chapter 11, "Command Descriptions.")

NOTE: The version number displayed in your identification message may vary from what is shown above.

When the system prompt and cursor appear, MS-DOS is ready to accept commands.

Your results may vary. In some cases, a program may run before the system prompt appears. This is accomplished by modifying your system through an AUTOEXEC.BAT file. For further information on automatic command entry through an AUTOEXEC.BAT file, refer to Chapter 5, "Command Features."

After MS-DOS is booted, the system prompt appears in the following form

```
A>
```

where A is the drive letter of the drive you booted.

Table 6.3. Possible Drive Name Combinations

NUMBER OF FLOPPIES	FLOPPY NAMES	WINCHESTER NAMES
1 or 2	A (and B)	C, D, E and F
3	A, B and C	D, E, F and G
4	A, B, C and D	E, F, G and H

Drives are named with letters A through H. Each letter is followed by a colon (:). This letter-colon combination is called a *drive name*.

The supported drive names are A, B, C, D, E, F, G, and H. A and B refer to your 5.25-inch disk drives. C and D can refer either to additional 5.25-inch disk drives or to Winchester disk partitions. E, F, G, and H are used to refer to Winchester disk partitions.

The drive name of the Winchester disk partitions is dependent on the number of floppy units. Winchester disk drive names begin with the first available drive name (after B) not assigned to a floppy disk drive name. Table 6.3 illustrates the three possible arrangements of floppy and Winchester disk drive names.

Error Messages

There are two categories of error messages that you may receive while using the Boot command. The first consists of those messages that are displayed by the operating system. The second category consists of those messages displayed by the MFM-150 monitor.

Error Messages Generated by the Operating System

`Bad or missing Command Interpreter`

EXPLANATION: The Command Interpreter (COMMAND.COM) is not recognized as residing on the disk. Replace the disk with one that contains the COMMAND.COM file, and reboot.

`I/O error`

EXPLANATION: A disk error occurred during the bootup sequence. This may indicate a defective disk or partition. Boot up from another disk or partition, or attempt to recover the disk or partition. To recover a floppy disk, run BACKUP, FORMAT, and RESTORE. To recover a Winchester partition, run BACKUP, DETECT, FORMAT, and RESTORE.

`No bootable partitions`

EXPLANATION: A partition has not been specified in the partition table as the default boot partition. Use an explicit Boot command and specify the unit and partition to boot from.

`No system`

EXPLANATION: An attempt was made to boot a disk that does not contain the IO.SYS and MSDOS.SYS files. Replace the disk with one that contains the system files, and reboot.

`Not a bootable partition`

EXPLANATION: An attempt was made to boot a partition that does not have a valid MS-DOS boot loader. Use the FORMAT command with the /S switch to create a bootable partition (FORMAT will copy the boot loader onto that partition). For other operating systems, consult your Operating System manual for the appropriate instructions on creating bootable partitions.

`Partition not formatted`

EXPLANATION: An attempt was made to boot a partition that has not been formatted by the MS-DOS FORMAT command. This can happen after you have used the PART utility and altered the partition.

`Unable to read boot code from partition`

EXPLANATION: The boot record on the specified partition is either not present or has developed a bad sector. Boot up from another drive. Then, back up the files on your partition to another disk and format the partition from which you were trying to boot up when the error message appeared. If this partition is totally inaccessible, back up all partitions and run the PREP utility. (For further information on running PREP, refer to Chapter 16, "PREP.") If this error message occurs after using PREP, contact your Technical Consultation or the dealer from whom you purchased your microcomputer for assistance.

`Unrecognized command in CONFIG.SYS`

EXPLANATION: A CONFIG.SYS file exists which contains information on system characteristics that MS-DOS does not recognize as applying to your operating environment.

Error Messages Generated by the MFM-150 Monitor

If you make an error when you are typing in your bootup command line and press RETURN, the message `^Invalid Command!` and the monitor prompt (`A>`) will appear. The caret (^) will point to within one character of where the error has occurred.

When you receive any of the following MFM-150 monitor error messages, you must enter either **CTRL-BREAK** to return to the monitor, **CTRL-ALT-DEL** to reattempt bootup, or **CTRL-ALT-INS** to reattempt bootup while preserving dynamic defaults.

Error Messages

```
+++ DISK ERROR:  Drive not ready! +++
+++ DISK ERROR:  Seek failure! +++
```

EXPLANATION: These errors are caused when an attempt is made to boot the operating system from a disk, and no disk has been inserted into the disk drive. Make sure that a disk has been placed into the disk drive, that it was inserted correctly, and that the drive door was closed properly.

```
+++ DISK ERROR:  Sector not found! +++
+++ DISK ERROR:  CRC error! +++
+++ DISK ERROR:  Invalid address mark detected! +++
```

EXPLANATION: These errors are caused when an attempt is made to boot the operating system from a disk, and the disk in the drive is either faulty or does not contain a copy of the operating system. This can be corrected by using another disk. If, however, these errors often occur in your system, consult your Technical Consultation or the dealer from whom you purchased your microcomputer for assistance.

Summary

The MFM-150 monitor can boot up from any drive or partition that is properly attached to the system. During the bootup sequence, the monitor searches the disk for the boot loader. The monitor loads the boot loader into memory; and the boot loader begins the loading of the operating system files (IO.SYS, MSDOS.SYS and COMMAND.COM) into memory.

The method you use to boot your microcomputer is determined by the SW-102 switch setting. The switch can be set to manually boot or to automatically boot either the floppy or Winchester disk.

The MFM-150 monitor allows dynamic defaults for all parameters of the bootup command line. These defaults are set at the last explicit Boot command entered. Thus, after resetting the system, you can enter an ambiguous command and achieve the same results as from your last explicit command.

Bootup completion can occur in a variety of ways, such as the appearance of the `Enter new date:` prompt, or the execution of an AUTOEXEC.BAT file. Either way, the system prompt (`A>`) appears when the operation is complete.

The system prompt contains the name of the drive that was booted. Drive names A, B, C, D, E, F, G, and H are supported. These names are divided between floppy drive names and Winchester drive names. Winchester drive names are dependent upon the number of floppy drive names assigned.

Chapter 7
Directory Features

Overview

In this chapter you will learn about the directory structure of MS-DOS version 2 and how it resembles a tree, with many branches (or paths) leading to files. Because MS-DOS version 2 utilizes what is called a *hierarchical* directory structure, the access to files and multiple levels of directories is somewhat intricate. Since directories can be created or eliminated under this directory structure, (in much the same way files were in versions of MS-DOS prior to version 2) special procedures and commands have been developed that deal specifically with directories.

The organization of the directory structure is based on the concept of a tree. As with a tree, MS-DOS version 2 has a *root*. This root is the foundation of the directory structure much as the root of a tree is its foundation and beginning.

The root directory is the foundation of all other directories in your system, just as all of the branches of a tree owe their existence to the root of that tree.

Directories do not just "grow" out of the root. They must be created by you, using special *directory* commands.

Implementation of the tree structure is relatively simple. The root directory is like the directory of the versions prior to MS-DOS version 2. Subdirectories of the root have a special attribute set, indicating that they are directories. The subdirectories themselves are files, although special ones.

Directory Organization

Versions of MS-DOS version 2 and above use what is known as a *hierarchical directory structure*. This means that there is a multilevel structure of directories and files, with a top to bottom order of precedence.

This hierarchical directory structure can be thought of as an inverted *tree*. The root of the tree is at the top and is the primary or beginning level of the directory structure. This *root* directory is the directory that is automatically created when you format a disk.

The tree directory structure grows as you create directories for groups of files. Within each new directory, files and subdirectories can be created or removed.

It is possible for you to *travel* around the tree-structured directory. For instance, to find any file within the system, you could start at the root and travel down any of the branches to the desired directory or file. Conversely, you can start where you are (that is, any subdirectory in the directory structure) and travel towards the root.

All files created under MS-DOS are given names. You assign the names, or the names are assigned by default through various utilities. The name is saved, along with other information such as date and time, in a directory. When you next request that file by name, MS-DOS searches through the directory by looking for a file that matches the name you requested. The DIR command causes MS-DOS to give you a list of all the files contained within a specified directory.

Directory Organization

With MS-DOS version 2 you can store the directory entries for 112 files in the root directory of a double-sided/double-density floppy disk and 64 files in the root directory of a single-sided/double-density floppy disk. The root directory of a Winchester disk partition can hold entries for 64-1024 files (depending on the size of the partition). Subdirectories can hold the entries for any number of files that you can fit on the disk or partition.

MS-DOS version 2 alleviates the fixed size constraint by allowing a disk to contain not one, but several directories, each of which is limited in size only by the capacity of the disk itself. MS-DOS version 2 imposes a structure over these directories, thus making large numbers of files easier to handle.

When a disk is formatted by MS-DOS version 2, a single directory is created. This directory is called the root, since it is the beginning of the file system. This is the same directory that was used by versions prior to MS-DOS version 2 and is created so as to maintain compatibility with disks created under previous versions of MS-DOS. Once you format the disk, you can instruct MS-DOS version 2 to create other directories. You can give these directories names just as you would ordinary files. MS-DOS version 2 must keep track of these directories just as it does files. Therefore, it places the names of the directories that you create in other directories. The directory that contains the name of another directory is said to be that directory's parent. Figure 7.1 shows an example of a typical directory structure.

Directory Organization

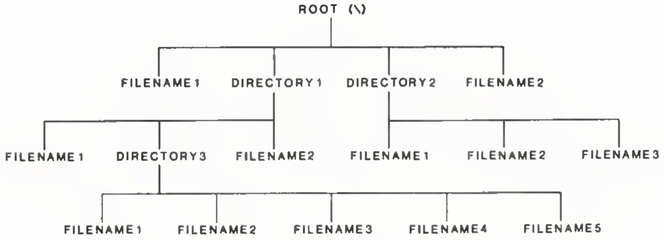


Figure 7.1 A Typical Directory Structure

The root directory has in it the names of two files and two directories. If you issued a DIR command for the root directory, only these four entries would be displayed. The directory "directory1" contains 3 names: two of them files and one of them another directory, "directory3." The other directory in the root, "directory2", contains only the names of three files. The parent directory of "directory3" is "directory1." "Directory3" contains the names of five files.

You will notice in the example that the files are given names which are the same as files in other directories. Only the files in a given directory have to be given unique names. Files that are in different directories may have the same names; since they are in different directories, there is no conflict.

You might find it easier to visualize this "multiple layer" directory structure by comparing it to our previous example of a tree. If you turn the diagram upside down, the root directory is at the base of the tree. The other directories in the tree are branches. Each branch can then branch out again (as in the case of "directory1" branching out to "directory3"). At the ends of the branches are the leaves, which are the individual files. When you use the ERASE (DEL) command and give it the name of directory, you actually are deleting all the files in that a directory.

Each disk that is in a drive has associated with it a *current working directory* of that disk, which may be changed with the CHDIR command. Unless you explicitly state that a file exists in a different directory (a different branch of the tree), MS-DOS version 2 searches for that file in the current working directory. When you issue the CHDIR command and change the current working directory, you can imagine transferring to a different branch of the tree.

When MS-DOS version 2 creates a directory for you, it places two special file names in that directory. This occurs for every directory you create, except the root directory. These two special file names are not actually files, but are simply names held in the directory's first two locations. These file names are "dot" (•) and "dot dot" (••). The •• entry is a synonym for the parent of this directory. Whenever you refer to •• you are referring to the parent of the current working directory. • refers to the directory itself. Referring to the name • is the same as referring to the current working directory.

Returning to the analogy of a tree, the directory name •• refers to the branch of the tree that the current branch grows out of. If you were climbing a tree, moving to the branch named •• means that you would climb closer to the root.

Directory Organization

Looking again at the example, assume that the current working directory is the root. To tell MS-DOS version 2 that you wish to erase the file "filename1" that is in "directory3", which is in turn in "directory1", you use a *path name*. A path name is simply a list of directories (sometimes ending with a file name) that MS-DOS version 2 follows in order to find a given file. The names of directories in a path name are separated by the \ character (backslash). The definition of path name is:

path name is a sequence of characters of the form:

```
[\] [directory] [\directory...] [\filename]
```

Where ***directory*** is any subdirectory in the system; and
filename is any valid file name for a disk file, including an optional file name extension.

NOTE: For all commands used in directory manipulation, the last entry in a path name must be a directory, not a file name. Also, the full path name of the current working directory cannot exceed 64 characters (not including ***d***: \).

The . and .. directory names may be used in lieu of the directory entries in a path name.

The current path in the example is directory1\directory3\filename1. This tells MS-DOS version 2 to "start at the current directory and find directory1. In directory1, find directory3. In directory3, find filename1."

Note that the example started with "start at the current working directory...". This is because MS-DOS version 2 always searches for files beginning in the current working directory. When you specify a path name of this type (beginning in the current working directory), it is known as a *relative path name*. In order to tell MS-DOS version 2 that the path is to begin at the root, the path name begins with a backslash. This type of path name is known as an *absolute path name*. Since the root is the current working directory in the example, an equivalent path name is:

\directory1\directory3\filename1

The opening backslash tells MS-DOS to begin following the path from the root directory.

In climbing a tree, you start at the root, then climb each branch in turn until you come to the leaf you are after. When MS-DOS is instructed to travel around its directories, it can either start at the root (if the path name is preceded by a backslash) or start at the current working directory.

Although most applications programs and languages written to run under previous versions of MS-DOS will function under MS-DOS version 2; these programs were written before the concept of multiple directories was introduced into MS-DOS. Because of this, these programs can only deal with files contained in the current working directory and do not accept path names to access other files.

Commands for Manipulating Directories

Since the directory structure of MS-DOS version 2 and above is dynamic (as opposed to the static directory structure of version prior to version 2), there have been special *directory* commands created to help you control the directory structure of MS-DOS version 2.

There are four commands which are used to control the directory structure of MS-DOS 2.0. These commands and their functions are:

PATH	used to designate a *path* for MS-DOS to search for any transient commands you might specify (that are not in the current working directory)
CHDIR	used to *change* the current working directory to a different one

Commands for Manipulating Directories

MKDIR	used to *make* a directory
RMDIR	used to *remove* a directory

With these four commands you can tell MS-DOS the correct path to search for any transient commands that are not located in your current working directory; change the current working directory to another directory; create (make) new directories from the root; and eliminate (remove) directories that are no longer needed.

Besides these four commands, the following command is also very important for working with the hierarchical directory:

COPY	this command can be used to copy (move) files from one subdirectory to another. This works the same way as copying files between disks. You should treat a subdirectory as you would a disk, for most command usage.

Refer to Chapter 11, "Command Descriptions," for a full discussion of all of these commands.

Paths and Transient Commands

Transient commands reside on-disk as program files. They must be read from the disk before they are executed.

When you are working with more than one directory, it is convenient to put all MS-DOS transient commands into a separate directory so they do not clutter your other directories. This is usually named the \BIN subdirectory, because it is a central storage directory for transient system commands. When you issue a transient command to MS-DOS, MS-DOS immediately checks your current working directory to find that command. MS-DOS will not search any other directory if the transient command file is not in the current directory and a path is not specified. You must tell MS-DOS in which directory these transient commands reside. This is done with the PATH command.

For example, if you are in a current working directory named \BIN\PROG, and all MS-DOS transient commands are in \BIN, you must tell MS-DOS to choose the \BIN path to find any transient commands. The command

```
PATH \BIN
```

tells MS-DOS to search your current working directory and then search the \BIN directory for all commands. You only have to specify this path once to MS-DOS during your terminal session. MS-DOS will now search in \BIN for the transient commands. If you want to know what the current path is, type the word PATH followed by a RETURN, and the current value of PATH will be printed. You may specify *more* than one path in the PATH command, and PATH will search each of them, in the order listed on the command line.

NOTE: The PATH command is used only for setting a path for transient *system* commands. It cannot be used to set a path for application program files. For users of application programs, this means that *all* supplementary files utilized by the application programs (or languages) must reside in the same directory. These types of files include (but are not restricted to) the following:

- text files
- data files
- overlays
- help files
- device drivers

For more information on the MS-DOS command PATH, refer to Chapter 11, "Commands Descriptions."

Paths and Resident Commands

Resident commands (the most commonly used) execute immediately because they are incorporated into the system processor. (For more information on resident commands, refer to Chapter 11, "Command Descriptions.")

Some resident commands can use paths. The following four commands COPY, DEL, DIR, and TYPE have greater flexibility when you specify a path name after the command.

The entry forms of these four commands are shown below.

`COPY` ***filespec*** [*d*:] [***pathname***] [/V]

If the second path name to copy is a directory, all files are copied into that directory.

`DEL` [*d*:] [***pathname***]
or
`ERASE` [*d*:] [***pathname***]

If the path name is a directory, all the files contained in that directory are deleted.

NOTE: The prompt, `Are you sure (Y/N)?` will be displayed if you try to delete the contents of a directory. Type Y to complete the command, or type N to abort the command. If you type Y, only the files in the directory are deleted. Directories can be deleted with the RMDIR command described later. The command

`DIR` [*d*:] [***pathname***]

displays the directory for a specific path.

The command

```
TYPE [d:][pathname]
```

will cause MS-DOS to display the file on your screen in response to the **TYPE [*pathname*]** command. However, you must specify a file name at the end of the path name, for this command to function.

Directory Access

The following sections discuss the commands used to manipulate directories, including: changing directories, creating directories, and removing directories.

Displaying Your Working Directory

All commands are executed while you are in your current working directory. You can find out the name of the current working directory by issuing the MS-DOS command CHDIR (Change Directory) with no options. For example, if your current working directory is \USER\JOE, when you type:

```
CHDIR
```

or

```
CD
```

your screen will display:

```
A:\USER\JOE
```

This is your current drive designation (A:) plus the current working directory (\USER\JOE).

If you now want to see what is in the \USER\JOE directory, you can issue the MS-DOS command DIR. The following is an example of the display you might receive from the DIR command for a subdirectory:

```
 Volume in drive A has no label
 Directory of A:\USER\JOE

.              <DIR>          1-09-84    10:09a
..             <DIR>          1-09-84    10:09a
TEXT           <DIR>          1-09-84    10:09a
FILE1   COM          5243     1-04-84     9:30a
4 File(s)    8376320 bytes free
```

A disk volume label was not entered when this disk/partition was formatted. (A disk volume label is an eleven character label which you may give to a disk at the time it is formatted). Note that MS-DOS lists both files and directories in this output. As you can see, Joe has another directory in this tree structure named TEXT. The '•' indicates the current working directory \USER\JOE, and the '••' is the shorthand notation for the parent directory \USER. FILE1.COM is a file in the \USER\JOE directory. All of these directories and files reside on the disk in drive A.

Because files and directories are listed together (see the previous display), MS-DOS does not allow you to give a subdirectory the same name as a file in that directory. For example, if you have a path \USER\JOE where JOE is a subdirectory, you cannot create a file in the USER directory with the file name JOE.

Changing Your Working Directory

Changing from your current working directory to another directory is easy under MS-DOS. Simply use the CHDIR (Change Directory) command and supply a path name. For example

```
CHDIR \USER
```
or
```
CD \USER
```

changes the current working directory from \USER\JOE to \USER. You can specify any path name after the command to "travel" to different branches and leaves of the directory tree. The command

```
CHDIR ..
```

will always put you in the parent directory of your current working directory.

Directory Creation

To create a subdirectory in your current working directory, use the MKDIR (Make Directory) command. For example, to create a new directory named NEWDIR under your current working directory, simply type:

```
MKDIR NEWDIR
```
or
```
MD NEWDIR
```

After this command has been executed by MS-DOS, a new directory will exist in your tree structure under your working directory. You can also make directories anywhere in the tree structure by specifying MKDIR and then a path name. MS-DOS will automatically create the '.' and '..' entries in the new directory.

To put files in the new directory, use the MS-DOS line editor, EDLIN. Files can also be created by applications programs (including word processors) and languages.

Directory Elimination

To delete a directory in the tree structure, use the RMDIR (Remove Directory) command. For example, to remove the directory NEWDIR from the current working directory, type:

```
RMDIR NEWDIR
```
or
```
RD NEWDIR
```

Note that the directory NEWDIR must not be the current working directory and must be empty except for the '•' and '••' entries. This will prevent you from accidentally deleting files and directories. If the directory you wish to remove contains anything other than the '•' and '••' entries, RMDIR will display an error message. You can remove any directory by specifying its path name. To remove the \BIN\USER\JOE directory, make sure that it has only the '•' and '••' entries; then, type:

```
RMDIR \BIN\USER\JOE
```

To remove all the files in a directory (except for the '•' and '••' entries), type DEL and then the path name of the directory. To delete all files in directory \BIN\USER\SUE, type:

```
DEL \BIN\USER\SUE
```
or
```
ERASE \BIN\USER\SUE
```

You cannot delete the '•' and '••' entries. They are created by MS-DOS as part of the hierarchical directory structure.

Example of an MS-DOS Tree Directory

To illustrate the use of the special directory commands that have been discussed, we will perform these commands on a model directory. Examine the example directory shown in Figure 7.2.

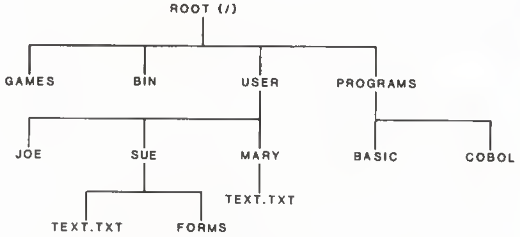


Figure 7.2. MS-DOS Tree Directory Example

The root directory is the first level in the directory structure. Although we refer to this as the root, it really has no name and may only be designated in command lines by using the \ (backslash) symbol. You have already seen how MS-DOS version 2 uses the MKDIR command to create subdirectories. In this example, four subdirectories of the root have been created. These include:

- a directory containing games, named GAMES
- a directory containing all transient commands, named BIN
- a directory containing subdirectories for all users of the system, named USER
- a directory containing programs, named PROGRAMS

Model Directory

In this example Joe, Sue, and Mary each have their own directory, which is a subdirectory of the USER directory. Sue has a subdirectory under the \USER\SUE directory for forms which is named FORMS. Sue and Mary each have a file named TEXT.TXT in their directories. Since these files are in separate directories, and thus unrelated, Sue and Mary can *both* use the name, TEXT.TXT for their individual files.

If you are working with other people who have their own subdirectories and files, or if you work concurrently on several projects at a time, knowing your way around the hierarchical directory structure becomes extremely useful. For example, you could get a list of all the files in Sue's FORMS subdirectory by typing:

```
DIR \USER\SUE\FORMS
```

Remember that the backslash character (\) must be used to separate directories from other directories and files.

You could also find out what files Mary has in her subdirectory, by typing:

```
DIR \USER\MARY
```

When you use hierarchical directories, you must tell MS-DOS exactly where the files are located within the directory structure. For example, both Mary and Sue have files named TEXT.TXT. If either of them wishes to access her respective file, she will have to tell MS-DOS which directory it is in. To accomplish this, the path name to the file must be given to MS-DOS. You will remember from the discussion of path names that a path name is a sequence of directory names followed by a simple file name, with each separated from the previous one by a backslash.

If a path name begins with a backslash, MS-DOS searches for the file beginning at the root (or top) of the directory tree. Otherwise, it begins at the current working directory and searches from there. If you specified the path name for Sue's file, TEXT.TXT, it would be:

```
\USER\SUE\TEXT.TXT
```

If you are in your current working directory, a file name and its corresponding path name may be used interchangeably. For example:

\	Indicates the root directory.
\PROGRAMS	Indicates the subdirectory under the root that contains program files.
\USER\MARY\FORMS\TAX	This is an absolute path name. This one is for a file named TAX in the subdirectory named FORMS which belongs to the USER named MARY.
USER\SUE	This is a relative path name; it names the file or directory, SUE in subdirectory USER of the current working directory. If the current working directory was the root (\), it would be named \USER\SUE.
FORMS	Indicates the name of a file or directory in the current working directory.

As explained earlier in this chapter, MS-DOS provides special shorthand notations for the current working directory and the parent directory (one level up from the current working directory). These notations (•, for the current working directory; and ••, for the parent) can be used instead of the full path name of the current working directory or the parent in a command line.

Model Directory

For example, if you type:

```
DIR ..
```

MS-DOS will list the files in the parent directory of your current working directory.

If you type:

```
DIR ..\..
```

MS-DOS will list the files in the *parent's* parent directory.

When you are working with more than one directory, it is convenient to put all MS-DOS transient commands into a separate directory, such as \BIN. That way you will always know where these important commands are located, and they will not take up space in your current working directory.

To use these commands, now that they are in their own directory, you will need to specify the PATH command. You will recall that the PATH command sets a path to these transient commands for each terminal session.

For example, if you are in a current working directory named \USER\SUE, and all of the transient commands are in \BIN, you must tell MS-DOS to choose the \BIN path to find any transient commands you need. You would do this by typing:

```
PATH \BIN
```

For more information on commands and how to use them, refer to Chapter 11, “Command Descriptions.”

File Attributes

The directories contain information on the size of files, the location of files on the disk, and the dates that they were created and updated.

Another system area, called the *File Allocation Table* (FAT), keeps track of the location of your files on the disk. It also allocates the free space on your disks so you can create files.

These two system areas enable MS-DOS to recognize and organize the files on your disks. The File Allocation Table is created on a new disk when you format it with the MS-DOS FORMAT command, and one empty directory is created, called the root directory.

MS-DOS version 2 sets certain attributes in files. These attributes show the command processor special information about a file. For example, one of the attributes that MS-DOS sets shows when a file is a directory. To MS-DOS a directory is really just a file with special attributes.

Summary

This chapter described the basic operation of the MS-DOS version 2 directory structure. Although MS-DOS is a very intricate and powerful set of programs, you should be able to manipulate the tree-like structure of MS-DOS after studying the explanations given in this chapter.

The directory manipulation commands are summarized in Table 7.1.

Directory Features

Summary

Table 7.1. Directory Manipulation Commands

COMMAND	PURPOSE	ENTRY FORM
COPY	Copies files	`COPY filespec [d:][pathname][/V]`
PATH	Sets MS-DOS search path	`PATH [pathname[;pathname]...]`
CHDIR or CD	Displays working directory; changes directories	`CHDIR [pathname]`
MKDIR or MD	Makes a new directory	`MKDIR [pathname]`
RMDIR or RD	Removes a directory	`RMDIR [pathname]`

Chapter 8
Input/Output Features

Overview

This chapter provides general information on how MS-DOS input and output functions are handled, how you may alter the default input and output used for most system commands and functions, how you may use a printer to selectively print output displayed on the screen, and how you may invoke alternate character fonts for your system.

Input/Output Sources and Destinations

All input and output functions under MS-DOS are accomplished by transferring information between *devices*. Devices include (but are not limited to) the keyboard, display screen, disk drives, printers, and modems and communications links. Devices may also be called *peripherals*.

The part of the operating system that keeps track of the devices connected to your system and that coordinates all transfers of information between devices is the Input/Output (I/O) Handler. (For more information about the I/O Handler, refer to Chapter 9, "System Component Features.")

For the I/O Handler to recognize and effectively use devices, a software interface that links a given physical device to the operating system must be provided. This software interface is called a *device driver*. A device driver is a file that contains all code required for the system to perform input/output functions using the device. A special header is included at the beginning of the file to identify it as a device driver, define the strategy and system interrupt entry points, and define various device attributes.

Input/Output Sources and Destinations

The device drivers provided with MS-DOS are of two types:

- Character device drivers that enable serial (or character-by-character) I/O such as that required by the keyboard, screen display, and printers. (ANSI.DVD is an example character device driver provided with MS-DOS; it is not loaded unless you specify it in a DEVICE= command in the CONFIG.SYS file. The CONFIG.SYS file is described in Chapter 9, "System Component Features.")

- Block device drivers that enable the transfer of blocks of data such as occurs when the system reads from or writes to a disk. The size of a block of data is defined by the device driver. Standard block device drivers define a *block* as 512 bytes of data.

Character device drivers (and their associated devices) are identified by a device name; only one physical device may be supported by one device driver at any given time. The names of character device drivers (device names) consist of from one to eight characters and conform to MS-DOS file-naming conventions and requirements.

Unlike character device drivers, each block device driver may support multiple devices of the same type. Thus, block devices are identified by single-letter names (drive names) rather than by the file name of the device driver.

Some character device driver names that are defined for use by MS-DOS are listed below, along with the device or device type to which each refers. These may be referred to as standard MS-DOS character device driver names. As your operating system software is shipped from the factory, the standard device names cannot be used as user-assigned file names or as application program file names. This restriction can be removed by including an AVAILDEV=FALSE command in the CONFIG.SYS file. (For information about the CONFIG.SYS file, refer to Chapter 9, "System Component Features.")

Input/Output Sources and Destinations

The standard character device driver names are:

- AUX or COM1—refers to the first equipped serial device, such as a serial printer or modem;
- CLOCK$—refers to a real-time clock device;
- COM2—refers to the second equipped serial device;
- CON—refers to the microcomputer terminal (keyboard and screen display);
- LPT2—refers to the second equipped parallel device;
- LPT3—refers to the third equipped parallel device;
- NUL—refers to the "null" device (Any output to the NUL device is discarded.); and
- PRN or LPT1—refers to the first equipped parallel printer or equivalent device.

The standard MS-DOS block device drivers support up to a total of eight disk drives and/or Winchester disk partitions that are identified by the device (drive) names A through H. The actual physical devices (if any) identified by these drive names may vary from system to system, depending upon the drives equipped. (For more information on the drive configurations supported by the standard MS-DOS block device drivers, refer to Chapter 6, "Bootup Features.")

Device names may be used as source and destination parameters in most command lines (such as for the COPY command) that require source and destination file specifications. For example, you could enter the command line

```
COPY CON filename
```

to create a file containing information input at the keyboard. (Refer to Chapter 11, "Command Descriptions," for more information about the COPY command.)

The NUL device is useful in this type of application. For example, suppose that you are using a program named LISTPROG that sorts and lists accounting and personnel information. Suppose that when you invoke the program, you must specify two parameters, one being the destination to which accounting information is to be output and the other being the destination to which personnel information is to be output. That is, to invoke the program, suppose you must enter a command line in the form

```
LISTPROG acctdest persdest
```

where ***acctdest*** is the file or device to which you want accounting information to be sent; and
persdest is the file or device to which you want personnel information to be sent.

If you wanted to run LISTPROG to obtain the accounting information it produces but did not want or need the personnel information, you could use the NUL device for the second parameter. That is, you could enter

```
LISTPROG PRN NUL
```

at the system prompt and press **RETURN**. The accounting information would be output to your printer, and the personnel information would be output to the NUL device (that is, it would be discarded). (Any output to the NUL device is discarded.)

Standard Input and Output

Since MS-DOS I/O is device-independent, a logical connection between the actual device driver (and, by extension, the actual device) used in any given I/O function must be established. This logical connection is provided by the system's use of redefinable STDIN (standard input) and STDOUT (standard output). That is, MS-DOS commands and functions always read the STDIN for input, and output to the STDOUT. The device drivers identified with the STDIN and STDOUT may not always be the same, however.

Input/Output Sources and Destinations

The standard device drivers provided by MS-DOS are described in the preceding section. Among the standard character device drivers, one is the default for the system's STDIN and STDOUT. This is the CON device, or your microcomputer terminal. The default STDIN is the microcomputer keyboard. The default STDOUT is the screen display.

Unless directed to do otherwise, MS-DOS will always look for and accept input you enter at the keyboard. Similarly, all output from MS-DOS commands and functions will be sent to the screen display unless you cause the system to do otherwise. The system's use of the default STDIN and STDOUT may be affected by your redefining the STDIN and/or STDOUT in two ways:

- input/output redirection, in which you redefine STDIN so that the system obtains input from a source other than the default STDIN (keyboard) or redefine STDOUT so that the system sends output to a destination other than the default STDOUT (screen display); and
- piping, in which you cause the system to use the output of one command as input to another command.

These methods are described under Input/Output Redirection and under Pipes and Filters, respectively, in this chapter.

Input/Output Redirection

Through the use of special command line parameters, you can easily cause MS-DOS commands, functions, and programs to obtain input from a file or device rather than from the default STDIN or to direct output to a file or a device other than the default STDOUT. The parameters used are

- <*source* when you want a command to obtain input from the specified *source* (a file or device),

- *>destination* when you want a command or program's output to be sent to the specified *destination* (a file or device), and
- *>>destination* when you want a command or program's output appended to the specified *destination* (normally an existing file).

Obtaining Input from a File or Device

You may use the < symbol to specify the input (that is, to redefine the STDIN) for any command or function that reads the STDIN for input. You may define the input to be a file or any readable device. The symbol and specified input are entered in a command line after the command, in the form

`<[d:][pathname]filename`

if you want the input obtained from a disk file instead of from the default STDIN (keyboard), or

`<device`

if you want the input obtained from a device other than the default STDIN. In these entry forms:

where ***d:*** is the drive name identifying the drive (if other than the default drive) in which the input file is located;
pathname is the directory path name identifying the directory (if other than the current directory of the default or specified disk) in which the input file is located;
filename is the file you want the command to read for input; and
device is the device name identifying the device from which you want the input obtained.

An example of an MS-DOS command line including input redirection is

```
SORT <MYFILE
```

where MYFILE is a file in the current directory of the default disk. (For information on the SORT command, refer to Chapter 11, "Command Descriptions.") In this example, STDIN is redefined to be the file MYFILE instead of the keyboard. If you entered this command at the system prompt and pressed **RETURN**, the contents of MYFILE would be sorted, line by line, into alphabetical order and displayed on the screen.

For another example, suppose you have created a program, MYPROG, that normally reads the default STDIN. Also, suppose that you have created a special application file that may be input to the program. Assume that MYPROG is in the current directory of the default disk and that the special input file INPUT.TXT is in the current directory of the disk in drive C. To execute the program by using the contents of INPUT.TXT as input instead of supplying program input via the keyboard, enter the following at the system prompt and press **RETURN**:

```
MYPROG <C:INPUT.TXT
```

Finally, suppose you are going to receive a listing via modem (device name COM1). To sort the listing as it is received, you could enter

```
SORT <COM1
```

at the system prompt and press **RETURN**. Then, as the transmission is received, it will be sorted before it is displayed on the screen.

NOTE: If an application program does not use MS-DOS function calls to perform input from STDIN and output to STDOUT, then I/O redirection will be ignored if you attempt to use it when invoking that program. In addition, when you use I/O redirection to specify a file as the source for input to a program, be *sure* that *all* required program input is contained in the file.

Input/Output Redirection

Sending Output to a File or Device

You may use the > symbol to specify an output destination (that is, to redefine the STDOUT) for any command or function that normally outputs to STDOUT. You may define the output destination to be a file or any writable device. The symbol and specified output are entered in a command line *after* the command name, in the form

>[*d*:] [*pathname*]***filename***

if you want the output written to a disk file instead of to the default STDOUT (screen display), or

>***device***

if you want the output sent to a device other than the default STDOUT. In these entry forms:

where ***d***: is the drive name (if other than the default drive) containing the disk on which the destination file is located;
pathname is the directory path name identifying the directory (if other than the current directory of the default or specified disk) in which the destination file is located;
filename is the file in which you want the output written; and
device is the device name identifying the device to which you want the output sent.

If you redirect output to a file, the file specified will be created if it does not already exist. If the file *does* exist, its contents will be overwritten. If you redirect output to a device, the *device* you specify must be a valid device name for a writable device. (Refer to Input/Output Sources and Destinations in this chapter.)

As an example, suppose you wished to store the directory of a data disk in a file on the program disk with which the data is used. If the program disk were in the default drive and the data disk were in drive C, you could accomplish this by entering

```
DIR C: >DATA.DIR
```

at the system prompt and pressing **RETURN**.

Normally, the DIR command causes directory information to be displayed on the screen (the default STDOUT). However, when you redirect STDOUT to a disk file in the above example, the directory is written to file DATA.DIR in the current directory of the default (program) disk.

As an example of redirecting output to a device, suppose you wished to make a hard copy listing of a data disk directory. You could accomplish this by redirecting the output of the DIR command to a printer that is properly connected and configured for your system. Suppose the data disk was in drive C. You could then enter the following at the system prompt and press **RETURN**:

```
DIR C: >PRN
```

The directory of the disk would then be printed as it was displayed on the screen.

NOTE: If an application program does not use MS-DOS function calls to perform input from STDIN and output to STDOUT, then I/O redirection will be ignored when invoking that program.

Appending Output to an Existing File

Using I/O redirection to write output to a file was described under Sending Output to a File or Device. When you use the method described there, any contents of the destination file (if it already exists) are lost; that is, existing file contents are overwritten by the new output. It is also possible to use I/O redirection to *append* output to the end of an existing file, such that existing file contents are not overwritten but new contents are added at the end of the file.

To append output to an existing file, use the >> symbol to specify an output file for any command or function that normally outputs to STDOUT. The symbol and specified output destination are entered in a command line *after* the command, in the form

>>[***d***:] [***pathname***]***filename***

where ***d***: is the drive name (if other than the default drive) containing the disk on which the destination file is located;
pathname is the directory path name identifying the directory (if other than the current directory of the default or specified disk) in which the destination file is located; and
filename is the file in which you want the output written. If the file does not already exist, redirecting output in this fashion will create the file.

You may also enter the >> symbol and destination after the command in the form

>>***device***

where ***device*** is a valid device name identifying the writable device to which you want the output sent.

Since you cannot literally append output to a device, the above entry form has the same effect as using the single right-angle bracket (>) to send output to a device.

Earlier in this chapter (under Sending Output to a File or Device), an example command line was shown for writing the directory of a data disk in drive C to a file named DATA.DIR on the default disk. That command line was DIR C: >DATA.DIR. Now, suppose the program disk is still in the default drive and that you have another data disk in drive D. Also, suppose you wish to append the directory of the disk in drive D to file DATA.DIR. You can accomplish this by entering

```
DIR D: >>DATA.DIR
```

at the system prompt and pressing **RETURN**. The directory for the disk in drive D will be appended to DATA.DIR, which already contains the directory of another data disk.

Pipes and Filters

A *filter* is a command that reads data from the STDIN, transforms or modifies it in some way, then writes the result to the STDOUT. Thus, the data received by the command has been "filtered."

Since filters can be used alone or together with other commands and filters in many different combinations, a few filters can take the place of a large number of specific commands. The MS-DOS filters and their individual functions are:

- CIPHER—Takes STDIN and encrypts or decrypts it (provided that the required keyword is entered), then outputs to STDOUT.
- FIND—Searches for a specified string of characters in the STDIN.
- MORE—Takes STDIN and displays it one screenful at a time.
- SORT—Sorts STDIN and outputs to STDOUT.

Pipes and Filters

These filters are described in more detail in Chapter 11, "Command Descriptions." Filters may easily be used with I/O redirection to accept input from files or devices or to send their output to files or devices. For example, the command

```
SORT <TRASH >TREASURE
```

sorts the contents of the file TRASH and writes the sorted contents to a file named TREASURE. (Note that, in this example, SORT and the file TRASH must be on the disk in the default drive. The new file, TREASURE, is created on the same disk.)

NOTE: If an application does not use MS-DOS function calls to perform input from STDIN and output to STDOUT, then you cannot use filters with that application.

Filters are also commonly used with *pipes*. A pipe (|) causes the STDOUT of one command or function to be used as the STDIN of another command or function. A pipe is used in a command line having the form

```
command1 | command2
```

where ***command1*** is the command whose output is to be "piped" to another command; and
command2 is the command that will receive as its input the output of *command1*.

NOTE: The spaces shown in the above command line entry form are optional.

Notice that, whenever a pipe is used, the output generated by the command to the left of the pipe (|) is input to the command on the right of the pipe. Also, you are not limited to two commands per command line when using a pipe. A *pipeline* (that is, a command line in which pipes are used) may consist of any number of commands as long as the input buffer limit of 127 characters is not exceeded.

Pipes may be used with any MS-DOS command as long as the command normally uses STDIN and STDOUT.

During pipeline operations, MS-DOS uses temporary files to hold the input and output data being piped from one command to another. The temporary files are created in the current directory of the default disk and have names in the form

%PIPE*n*.$$$

where *n* is an integer value.

Normally, you will never see these temporary files in your disk directory unless an error occurs during the operation being performed and execution is halted. MS-DOS erases the temporary files when piping has been successfully completed.

Some examples of pipelines are provided in the paragraphs that follow.

Suppose you wished to obtain a printed copy of the directory of a commonly-used data disk that was currently in drive B. Also suppose that you wanted the directory alphabetized for ease of reference. To obtain the desired directory listing, you would enter

```
DIR B: | SORT >PRN
```

at the system prompt and press **RETURN**. The alphabetized directory for the disk in drive B would be printed.

You could use a similar command to write the sorted directory to a disk file. Suppose you wanted the directory stored in a file named DIREC.FIL in subdirectory REF on the disk in the default drive. Assuming that \REF already exists, you could accomplish this by entering

```
DIR B: | SORT >\REF\DIREC.FIL
```

at the system prompt and pressing **RETURN**.

As a final example, suppose you wanted to sort and display the lengthy directory for a Winchester disk partition designated by drive name C. To simplify your review of the directory, you could use a combination of pipes and filters as shown in the following command line:

```
DIR C: | SORT | MORE
```

If you enter this command at the system prompt and press **RETURN**, MS-DOS will first sort the directory alphabetically, then display it one screen at a time. At the bottom of each screen of the directory, the system will display

```
-- MORE --
```

to indicate that there is more output (directory entries, in this example) to be displayed. You may press any alphanumeric key to display the next screen of information. When the entire sorted directory has been displayed, the system prompt will be displayed again.

Printing Screen Displays

MS-DOS provides you a number of methods by which to print information that is displayed on the screen. There are two ways to turn printer echo on and off, and a group of utilities for selectively dumping full screens of information to a printer. (One utility is a system-resident function; others are file-resident and printer specific, and must be loaded into memory by execution of a PSC command line.)

Printer echo, once turned on, is a continuous process. Until printer echo is turned off, all alphanumeric information displayed on the screen (including keyboard input) is printed character by character as it is displayed. With the screen dump printing capabilities, you may selectively print screen displays. That is, one full screen of information is printed at a time, subject to your invoking the print function for any given screen.

Some of the methods support the printing of graphics characters; others do not. The printer echo function and the system-resident screen dump capability support the printing of alphanumeric (that is, ASCII-only) screen displays. The file-resident screen dump utilities support the printing of all information displayed on the screen (including graphics and special characters as well as ASCII characters) when the system is in graphics mode. When the system is not in the graphics mode, only ASCII characters are printed.

A general description of all supported methods is provided in the paragraphs that follow. In order to use any of the methods, you must have a printer connected to your system, and your system must be properly configured for the specific printer.

Invoking Printer Echo

Of all the methods supported by MS-DOS for printing screen displays, printer echo is perhaps the simplest. Once you turn printer echo on, every line that is displayed on the screen is echoed to your printer (device name PRN or LPT1). Each line is printed as it is displayed. Printer echo is a continuous function; it stays on until you turn it off. Only ASCII characters are printed; graphics characters are not printed.

You may turn printer echo on and off in one of two ways, both of which are control key functions. In function, the two methods are identical to each other. One way you can turn printer echo on is to press CTRL-PRTSC. Pressing CTRL-PRTSC a second time turns printer echo off. The second way you can turn printer echo on is to press CTRL-P. Pressing CTRL-P a second time turns printer echo off.

Since the control key functions used to turn printer echo on and off are simple toggles and do not affect other MS-DOS functions, you can turn printer echo on and off as desired during any work session.

Selectively Printing Screen Displays

In addition to printer echo, MS-DOS provides you the capability to print screen displays selectively, such that you can print one full screen of information at a time.

The capability to selectively print (or dump to the printer) screen displays that consist of ASCII characters is inherent in the microcomputer itself. A group of utilities that are loaded into memory by execution of a PSC command provide screen dump printing capability for graphics screen displays. These utilities are printer-specific. To use any of the screen printing capabilities, you must have a printer properly connected to your system, and your operating system must be configured for the specific printer.

Printing ASCII Screen Displays

To print any ASCII screen display, a system-resident control key function may be invoked. To invoke this function, press **SHIFT-PRTSC.** All ASCII information displayed on your screen at the time you press SHIFT-PRTSC will be printed. After one full screen display has been printed, printing stops. No other information is printed until you press SHIFT-PRTSC again. Thus, you may print one, some, or all of the screen displays during any work session. (Note, however, that if you wish to print *all* information displayed on the screen, it is simpler and more efficient to use printer echo.)

Printing Screen Displays with Graphics or Special Characters

All of the printing capabilities described thus far are for ASCII characters only. MS-DOS also provides PSC (print screen) utilities that support the printing of screen displays that include graphics and special characters as well as ASCII characters. The PSC utilities are provided for specific printers, including the following:

- Epson MX-80 Printer with graphics option,
- IDS Prism Printer,
- MPI Printer,
- Okidata Printer,
- Printek 920, and
- Transtar 315 Color Printer.

For detailed information on the PSC utilities, refer to Chapter 11, "Command Descriptions."

NOTE: In order for you to use any of the PSC utilities to print graphics screen displays, your system must be in the graphics mode. (If your system is not in the graphics mode, only ASCII characters will be printed, even if a PSC utility has been loaded into memory.) Most application programs that support or use graphics will already be in the graphics mode. Otherwise, you may enter the graphics mode from BASIC by issuing the SCREEN statement.

The general sequence for using a PSC utility is as follows:

- At the system prompt, enter a PSC command line for the graphics-capable printer you will be using. The system then loads the specified PSC utility into memory, where it remains until you replace it with another PSC utility, or until you turn off or reboot your system. Typically, a PSC command line need be entered only once in a session.

- Begin or continue your work session in graphics mode. Whenever you want to print a screen display, press SHIFT-PRTSC. All information currently displayed on the screen (including graphics, ASCII, and special characters) will be printed.

- After one full screen of information has been printed, printing stops. No other information is printed until you press SHIFT-PRTSC again.

Notice that, once a PSC utility has been loaded into memory, you may selectively dump graphics screen displays to the printer at any time you wish while you are in the graphics mode. If you exit the graphics mode (that is, if you change to a non-graphics video mode), the system-resident SHIFT-PRTSC function for ASCII-only characters will be in effect. (See the Printing ASCII Screen Displays section of this chapter.) Note, however, that the PSC utility you loaded with a PSC command line remains in memory and may be used if you return to the graphics mode.

Alternate Character Fonts

For those with programming experience, it is possible to create alternate character fonts for use under MS-DOS. When you operate your microcomputer in the graphics mode, a total of 256 characters may be used. However, only the first 128 characters are defined in video ROM; these are characters in the standard Z-100 PC character set. The second 128 characters may be defined as desired in a memory table of up to 1K bytes in size; each character is represented by 8 bytes of graphic information. Then, to use the alternate character font, interrupt vector 1FH must be changed to point to the table in which the font is defined. (Interrupt vector 1FH is initialized to 0:0.)

Your microcomputer must be in the graphics mode in order for you to use alternate character fonts.

Summary

All MS-DOS input and output functions are accomplished by transferring information between devices. The software interface between the MS-DOS I/O Handler and various physical devices is provided by character and block device drivers. The internal logical connection between the I/O Handler and the device drivers is provided by the system's use of STDIN (standard input) and STDOUT (standard output).

The default STDIN is the microcomputer keyboard; the default STDOUT is the screen display. You may temporarily redefine the STDIN and the STDOUT in order for the system to accept input from a file or device other than the keyboard and in order for the system to output to a file or device other than the screen display. You may do this by entering a command line in which you use I/O redirection and/or pipes. The STDIN and STDOUT may be redefined to be any valid file name or device. If you redefine STDIN to be a device other than the default STDIN, the device you specify must be a readable device. If you redefine STDOUT to be a device other than the default STDOUT, the device you specify must be a writable device.

MS-DOS provides you a number of methods by which you can obtain a hard copy of information displayed on the screen. You can invoke printer echo, selective screen dumps of ASCII characters, or selective screen dumps of all characters (including graphics and special characters as well as ASCII characters) for a specific group of printers. Printing graphics information requires you to use a PSC utility; the other print screen capabilities are system-resident.

Chapter 9
System Component Features

Overview

MS-DOS is an operating system that consists of several main components. Each of these components has specific functions and specific locations when recorded on disk media and when loaded into microcomputer memory.

This chapter explains what each system component does and where it does it. These explanations can help you to understand the invisible operations that the system performs while it is helping you to perform your microcomputer activities.

This chapter discusses both the generic components and the MS-DOS components of this operating system. The generic components are those components that are common to all microcomputer operating systems—including MS-DOS. The MS-DOS components are those components specific to MS-DOS.

Generic Components

Apart from all of its special features, accompanying utilities, and enhancements, MS-DOS has aspects common to most microcomputer operating systems. Most microcomputer operating systems have the following five main generic components:

- Input/Output Handler – managing system input and output,
- File Handler – managing the file system,
- Memory Handler – managing allocation of microprocessor memory,
- Executive – loading programs and causing the microcomputer to execute the instructions in the program, and

- Boot Loader – beginning the process of loading the other system components (inactive after all components have been loaded).

NOTE: To perform the functions of these five generic components, this operating system uses four MS-DOS components. These MS-DOS components; MSDOS.SYS, IO.SYS, COMMAND.COM and MS-DOS Boot Loader, are discussed in the next section of this chapter.

Input/Output Handler

The Input/Output Handler directs the movement of information within the system. It keeps track of what peripheral devices are connected to the system and where they are connected. Using this information and the directives of the Executive, the Input/Output Handler coordinates all transfers of information between devices connected to the microcomputer. This coordination permits each device to send and receive information to and from various locations.

The functions of the Input/Output Handler are performed in this system by the two MS-DOS components MSDOS.SYS and IO.SYS.

File Handler

The File Handler handles all file manipulation requests. It packs data into groups and records these groups in a structure called a *file*. The File Handler also keeps track of where individual files are located on the disk and retrieves information from these files when the information is requested. It records the files' names and locations in a directory that is written to disk.

Each time new disks are inserted in the disk drives, the File Handler must read the disks' directories to learn what files are stored on them and where they are located on the disks.

The information that is stored under the control of the File Handler is considered *nonvolatile (or long-term) storage* because the files remain stored on the disks, even when the system is reset or turned off.

The functions of the File Handler are performed in this system by the MS-DOS component MSDOS.SYS.

Memory Handler

The Memory Handler allocates the microcomputer's Random Access Memory (RAM) to the software and/or data that is loaded to perform a task.

The Memory Handler also recovers memory by loading new software and/or data into the location of old material (overwriting the old material) when the old material is no longer needed.

The RAM that is managed by the Memory Handler is considered *volatile (temporary) storage* because material stored in RAM is destroyed when the system is reset or turned off.

The functions of the Memory Handler are performed in this system by the MS-DOS component MSDOS.SYS.

Executive

The Executive is at the hub of the operating system organization. The Executive controls the operating system as a whole, directs functions, and makes sure that the correct programs are executed. The Executive acts as the interface between user input and the other generic components of the operating system because the Executive interprets the commands you enter.

Functional Components

The Executive responds to your request for a program. When you request a program, the Executive directs the appropriate operating system component to:

- locate the program requested,
- copy the program into a workspace in system memory, and
- begin execution of the program's instructions.

The functions of the Executive are performed in this operating system by the MS-DOS component COMMAND.COM.

Boot Loader

The boot loader helps load the rest of the system into computer memory during bootup.

The boot loader is recorded on the first sector of bootable storage media. It must contain enough information to access the booted media and find other components of the system.

After the rest of the system is loaded into memory, the boot loader might not be used any more. With some systems, the boot loader might even be overwritten after it has performed its function.

The functions of the Boot Loader are performed in this system by the MS-DOS component called the MS-DOS boot loader, which is described later in this chapter.

MS-DOS Components

This operating system is divided into the following four MS-DOS components:

- MSDOS.SYS – hardware-independent director of activities,
- IO.SYS – hardware-dependent communicator with devices,
- COMMAND.COM – hardware-independent command interpreter, and

- MS-DOS Boot Loader – hardware-dependent mover of system from disk to memory.

This section describes the properties of each of these MS-DOS components. Since the first three components are stored in a disk file, this section refers to these components by their file names.

MSDOS.SYS Properties

MSDOS.SYS is considered the director of the system. It issues logical (or generalized) instructions for work to be done. Its instructions do not include specific information about what hardware should do the work or how. Therefore, MSDOS.SYS is considered to be *hardware-independent* because it performs its functions without regard for the physical characteristics or location of the hardware.

Because of its hardware-independence, MSDOS.SYS can be used on microcomputers with any peripheral hardware.

MSDOS.SYS provides a high-level interface for user programs. It consists of file management routines, data blocking/deblocking for the disk routines, and a variety of built-in functions easily accessible by user programs. When these function routines are called by a user program, they accept high-level information via register and control block contents. Then (for device operation) they translate the requirement into one or more calls to IO.SYS to complete the request.

MSDOS.SYS performs all file management and disk management activities. MSDOS.SYS also performs part of the input/output management activities. (See the section of this chapter on Generic Components.)

IO.SYS Properties

IO.SYS can be considered the worker of the system. It issues physical, or specific, instructions for work to be done. The instructions it issues include specific information about what hardware should do the work and how the hardware should do the work. For this reason, IO.SYS is considered to be *hardware-dependent.*

IO.SYS must receive the logical, or generalized, instructions from MSDOS.SYS and translate them to instructions that include the specific physical characteristics of the hardware.

For instance, MSDOS.SYS might give IO.SYS an instruction for some data to be output through the "CON logical device." IO.SYS knows that "CON logical device" is MSDOS.SYS's way of referring to the "console," which consists of a keyboard and a video display screen. Therefore, IO.SYS would output the data through the console's video screen. IO.SYS would also have to take into consideration the screen characteristics such as its display speed (baud rate) or its display width (number of columns in a displayed line.)

(For more information about logical devices and their relationships with hardware, refer to Chapter 8, "Input/Output Features.")

Because of its hardware-dependence, a particular implementation of IO.SYS can be used on only a specific type of microcomputer and with only a certain type of peripheral hardware. IO.SYS must be modified whenever it is used on hardware whose characteristics are significantly different from those of the last hardware environment in which it was used.

IO.SYS performs part of the input/output management activities. (See the section of this chapter on Generic Components.)

COMMAND.COM Properties

COMMAND.COM performs all of the functions described in this chapter's explanation of the Executive generic component.

In general, COMMAND.COM acts as the interface between user input and the other components of the operating system. COMMAND.COM provides this interface by interpreting entered commands.

COMMAND.COM is sometimes referred to as the "command interpreter," the "console command processor," or the "shell." MS-DOS allows you to substitute COMMAND.COM with a different program that also performs the Executive functions. To make this substitution, refer to the section, Supplementing the MSDOS.SYS Component of this chapter.

NOTE: For information on entering COMMAND.COM as a transient command at the system prompt, refer to Chapter 11, "Command Descriptions."

COMMAND.COM consists of a resident, initialization, and transient portion. Each of these portions is described below.

Resident Portion

The resident portion includes the software that enables the system to load programs, to handle device errors, and to interrupt program execution when you enter CTRL-BREAK or CTRL-C. This portion remains resident in memory from the time you boot up the system until the time you reset the system. It resides in memory immediately below MSDOS.SYS and the MSDOS.SYS data area. The resident portion also includes software that causes reloading of the transient portion of COMMAND.COM if it has been overwritten by a loaded transient program.

Physical Components

Initialization Portion

The intialization portion is located beneath the resident portion and is given control during startup. This section contains the AUTOEXEC.BAT file processor setup routine. The initialization portion determines the segment address at which programs can be loaded. It is overwritten by the first program that COMMAND.COM loads because it is no longer needed.

Transient Portion

The transient portion of COMMAND.COM includes all of the MS-DOS resident commands, such as DIR, EXIT, PATH, and GOTO. (For a complete list of resident commands, refer to Chapter 5, "Command Features.") This portion also contains the resident command processor and the batch processor.

The transient portion is loaded at the high end of memory. This portion contains the internal command processors and the batch file processor. It also causes the display of the system prompt, reads commands entered through the keyboard (or batch file), and causes such commands to be executed.

When a transient command (other than COMMAND.COM) is loaded for execution, this command may overwrite the transient portion of COMMAND.COM if it needs the memory space. Therefore, the transient portion of COMMAND.COM may have to be reloaded from a disk or partition containing the COMMAND.COM file after the transient command is executed.

The relationship between the transient and resident portions of COMMAND.COM and the transient and resident commands can be confusing. It is important to understand that the resident commands do not need to be in memory while a transient command is executing because transient commands are designed to perform their tasks without assistance from resident commands. Therefore, the resident commands are stored in the portion of COMMAND.COM that can be overwritten in memory when a transient command needs more memory in order to load or execute. This portion of COMMAND.COM is known as the transient portion because it can be overwritten and reloaded while the other components of COMMAND.COM remain resident in memory.

MS-DOS Boot Loader Properties

The MS-DOS boot loader is a program that participates in the act of loading the other system components from a drive to microcomputer memory during bootup.

The boot loader is not contained in a file, but recorded on the first sector of every disk or partition that is formatted by the MS-DOS FORMAT command. When bootup begins, the boot loader is loaded into microcomputer memory. When in memory, the boot loader causes the IO.SYS system component file to be loaded from the booted disk or partition.

The boot loader contains information about disk format so that it knows how and where to find the system files that it must load. When IO.SYS has been loaded, the boot loader transfers control to IO.SYS. Then the boot loader is no longer used in memory. The instructions issued by the boot loader include specific information about what hardware should do the work and how. For this reason, the MS-DOS boot loader is considered to be *hardware-dependent*.

The boot loader used by MS-DOS performs all of the functions described in this chapter's explanation of the "Boot Loader" generic component.

Disk Locations of Components

The operating system MS-DOS components can be recorded on a disk or partition by the transient commands FORMAT and SYS.

Disk Locations of Components

You can record all four MS-DOS components by entering a FORMAT command with the "/S" switch. This command will copy the components to the disk or partition in the following sequence:

Boot Loader
IO.SYS
MSDOS.SYS
COMMAND.COM

FORMAT will copy the boot loader to the disk or partition whether or not you specify the /S switch in the FORMAT command line.

FORMAT records these components in the order listed above.

FORMAT records the boot loader component on the first sector (at sector 0 of track 0 on side 0) of a floppy disk, or on the first sector of a Winchester disk partition.

On the sectors following the boot loader, FORMAT will record a reserved area, an initial copy of the File Allocation Table (FAT), a second copy of the FAT, the root directory, and the data area.

FORMAT will record the IO.SYS and MSDOS.SYS files beginning on the first sectors of the data area. FORMAT gives these files hidden file status so that they cannot be renamed or moved from their location.

NOTE: The hidden status given to the IO.SYS and MSDOS.SYS files also prohibits you from viewing these files with the DIR command or from manipulating them with any other command that is provided with MS-DOS.

When the boot loader is loaded into memory during bootup, it will look at the first directory entry to find and load IO.SYS. If the IO.SYS entry is not found at this location, then the following error message will be displayed:

```
No system
```

This error message can occur if you try to boot up a disk or partition that was formatted by entering a FORMAT command line without the /S switch.

The COMMAND.COM file can be recorded on any part of the data area that is available beyond the MSDOS.SYS file. COMMAND.COM is not given hidden file status. Therefore, you can view it with the DIR command, copy it with the COPY command, or delete it with the DEL command.

If the COMMAND.COM file (or another command interpreter specified in the CONFIG.SYS file) does not reside on a disk or partition when you try to boot up this media, the following error message is displayed:

```
Bad or missing Command Interpreter
```

The purpose of the SYS command is to record three of the MS-DOS components onto a formatted disk or partition in the following sequence:

Boot Loader
IO.SYS
MSDOS.SYS

SYS gives the IO.SYS and MSDOS.SYS files hidden file status.

The COMMAND.COM file is not recorded on the disk or partition by SYS. If you use SYS, you should record COMMAND.COM (or another suitable command interpreter) on the disk or partition with the COPY command. Because COMMAND.COM can be recorded anywhere on the disk or partition (beyond the sectors occupied by MSDOS.SYS), you can copy other files to the disk or partition before copying COMMAND.COM to the disk or partition.

NOTE: The commands FORMAT, SYS, DIR, COPY, and DEL are explained in alphabetically-arranged sections of Chapter 11, "Command Descriptions."

Memory Locations of Components

The operating system MS-DOS components are each allocated specific areas in Random Access Memory (RAM) when MS-DOS has been booted and the system prompt is being displayed.

Figure 9.1 shows the relative arrangement of the components at the point when the system prompt is displayed. In Figure 9.1 the components at the top are considered to be in "low memory" or RAM areas with low addresses. The components at the bottom are considered to be in "high memory" or RAM areas with high addresses.

Interrupt vector table
IO.SYS – MS-DOS interface to hardware
MSDOS.SYS – MS-DOS interrupt handlers, service routines
MS-DOS buffers, control areas, and installed device drivers
Resident portion of COMMAND.COM – Interrupt handlers for the Terminate Address, the CTRL-BREAK Exit Address, and the Fatal Error Abort Address and code to reload the transient portions
Transient command loading area – (.COM or .EXE file)
Transient portion of COMMAND.COM – Command interpreter, resident commands, batch processor

Figure 9.1. Memory Locations of MS-DOS Components

Component Behavior During Bootup

When you boot up successfully, your hardware, firmware, and software participate in a coordinated effort to put you in control of your microcomputer environment. During this effort, several steps occur automatically and invisibly.

1. Bootup begins. (Instructions on performing a bootup activity can be found in Chapter 6, "Bootup Features.")

2. If you are booting up from a floppy disk, then skip ahead to step 5.

   If you are booting up from a Winchester disk partition, then proceed to step 3.

3. The monitor program loads the boot record from the first sector of the Winchester disk into memory, and transfers control to this boot record. (Refer to your microcomputer hardware documentation for more information about the monitor program. Refer to Chapter 17, "PART," for more information about the boot record.)

4. The boot record loads the boot loader program from the specified or default boot partition into memory, and transfers control to the boot loader.

   a. If the specified or default boot partition contains MS-DOS version 2, then skip to step 6.

   b. If the specified or default boot partition contains an operating system other than MS-DOS version 2, then this other system finishes the bootup activity in its own fashion.

5. The monitor program loads the boot loader program from the floppy disk into memory and transfers control to the boot loader. (Refer to your microcomputer hardware documentation for more information about the monitor program.)

Component Behavior During Bootup

6. The boot loader program loads the IO.SYS file from the disk or partition into memory and transfers control to IO.SYS.
7. IO.SYS initializes the hardware devices and loads MSDOS.SYS.
8. IO.SYS transfers control to its SYSINIT module.

NOTE: SYSINIT is a software module linked to the end of IO.SYS for the purpose of relocating and initializing MSDOS.SYS.

9. SYSINIT relocates itself in memory.
10. SYSINIT relocates the MSDOS.SYS program in memory.
11. The SYSINIT routine initializes the tables of the MSDOS.SYS program.
12. MSDOS.SYS displays an identification message in the following form:

```
MS-DOS version 2.11
Copyright 1981,82,83 Microsoft Corp.
```

NOTE: The version number displayed by your software may differ from that shown in examples of this manual.

13. SYSINIT looks for the CONFIG.SYS file.
    a. If the CONFIG.SYS file exists in the root directory of the disk or partition, then SYSINIT modifies MSDOS.SYS according to the subcommands listed in the CONFIG.SYS file. Then SYSINIT proceeds to step 14.
    b. If the CONFIG.SYS file does not exist in the root directory of the disk or partition, then SYSINIT proceeds to step 15.

NOTE: The CONFIG.SYS file is described in the section, Supplementing the MSDOS.SYS Component.

14. SYSINIT looks through the root directory of the default or specified drive for the command interpreter (or shell) specified in the CONFIG.SYS file, or for COMMAND.COM if no shell was specified.

    a. If COMMAND.COM exists in the root directory of the default or specified drive, and if a CONFIG.SYS file did not specify a different shell, then SYSINIT loads and transfers control to COMMAND.COM.

    b. If COMMAND.COM does not exist in the root directory of the default or specified drive, or if a CONFIG.SYS file specified a different shell that does not exist in the root directory of the default or specified drive, then the `Bad or missing Command Interpreter` error message will be displayed.

    c. If the CONFIG.SYS file specified a shell other than COMMAND.COM, and if that shell exists in the root directory of the default or specified drive, then SYSINIT will load and transfer control to that shell. (A shell other than COMMAND.COM might not behave the same as COMMAND.COM in the operations described throughout this manual.)

15. COMMAND.COM looks for the AUTOEXEC.BAT file.

    a. If the AUTOEXEC.BAT file exists in the root directory of the booted disk or partition, then COMMAND.COM triggers sequential execution of the commands listed within the AUTOEXEC.BAT file.

    b. If the AUTOEXEC.BAT file does not exist in the root directory of the booted disk or partition, then COMMAND.COM triggers sequential execution of the DATE and TIME resident commands.

(Refer to Chapter 5, "Command Features," for more information about the AUTOEXEC.BAT file.)

16. COMMAND.COM displays the system prompt and waits for you to enter a command line through the console keyboard.

Component Behavior After Bootup

When all MS-DOS components have been loaded into memory (by bootup), your microcomputer is either waiting for entry of a command at the system prompt or executing a command.

The ways in which you can edit and enter command lines are explained in Chapter 5, "Command Features." The activities that occur during command execution are documented in the command guides (Parts III, IV, and V) of this manual.

However, this section describes the activities that generally occur during the brief time between the completion of command line entry and the beginning of command execution.

Command Interpretation

The following sequence of steps shows how MS-DOS generally interprets and responds to an entered command line.

1. Command line is entered at a system prompt.
    a. If the command line was entered by you directly through the console keyboard, then command entry was completed when you pressed the RETURN key.
    b. If the command line was entered from a batch file, then command entry was completed when the RETURN entry (invisibly recorded in the batch file) was automatically read from the batch file.
2. The entered command line is received by the COMMAND.COM system component.

NOTE: If the COMMAND.COM file was not loaded during bootup, then the command interpreter that was loaded will receive the entered command line. A command interpreter other than COMMAND.COM might *not* behave the same as COMMAND.COM in the operations described throughout the remaining steps.

3. COMMAND.COM scans the command line and compares the command function with the resident command functions (such as DIR, PROMPT, or GOTO) that are stored within COMMAND.COM.

   a. If the entered command function matches a resident command function, then execution of the command begins immediately.

   b. If the entered command function does not match a resident command function, then proceed to step 4.

4. COMMAND.COM examines the directory of the disk or partition in the default drive or in the drive specified by the command line.

   a. If the default or specified drive contains a single transient command file (with a .COM, .EXE, or .BAT extension) that matches the entered command function, then this transient command file is loaded and command execution begins.

   b. If the default or specified drive contains more than one transient command file that each match the entered command function, then one of these transient command files will be loaded and executed according to the following priorities:

      1) A transient command with the .COM file name extension (such as CHKDSK.COM or DEBUG.COM) will be loaded and executed—even if it exists in a drive containing a .EXE file and/or a .BAT file with the same primary file name.

2) A transient command with the .EXE file name extension (such as FC.EXE or FIND.EXE) will be loaded and executed—even if it exists in a drive containing a .BAT file with the same primary file name.

c. If the default or specified drive does not contain a transient command file that matches the entered command function, then the error message `Bad command or file name` will be displayed and followed by the system prompt.

Command Execution Priority

A command line function can refer to any of four different kinds of commands (a resident command, a .COM file, a .EXE file, or a .BAT file). Therefore, if an entered command line function matches more than one type of command that is resident in the system or recorded on the disk, MS-DOS must determine which of these commands will be executed.

The COMMAND.COM system component determines that the command to be executed is the highest available command in the following list of execution priorities:

- Resident command (resides in memory and has no file name extension)
- Transient command with the .COM file name extension
- Transient command with the .EXE file name extension
- Transient command with the .BAT file name extension

For instance, if the following hypothetical command line were entered at a system prompt:

```
PROGRAM C:THISFILE.OBJ/K/R
```

then COMMAND.COM will scan the command line and find the function "PROGRAM." COMMAND.COM will first try to match PROGRAM with its list of resident command functions, as shown:

BREAK
CD
CHDIR
CLS
COPY
CTTY
DATE
DEL
DIR
ECHO
ERASE
EXIT
FOR
GOTO
IF
MD
MKDIR
PATH
PAUSE
PROMPT
RD
REM
REN
RENAME
RMDIR
SET
SHIFT
TIME
TYPE
VER
VERIFY
VOL

Then, if the entered command line function PROGRAM does not match any of the valid resident command functions (as is the case), COMMAND.COM will try to find a transient command file on the disk or partition in the default drive.

If COMMAND.COM finds more than one transient command with the PROGRAM primary file name in the drive, it will select one of them to be loaded and executed. Table 9.1 presents several possible scenarios where one or more transient command files are recorded on the disk and indicates which of them COMMAND.COM would select.

Table 9.1. Transient Command Execution Priorities

EXECUTABLE FILES WITHIN DEFAULT DRIVE	FILE SELECTED BY COMMAND.COM FOR LOADING AND EXECUTION
PROGRAM.COM	PROGRAM.COM
PROGRAM.EXE	PROGRAM.EXE
PROGRAM.BAT	PROGRAM.BAT
PROGRAM.COM PROGRAM.EXE	PROGRAM.COM
PROGRAM.EXE PROGRAM.BAT	PROGRAM.EXE
PROGRAM.COM PROGRAM.EXE PROGRAM.BAT	PROGRAM.COM

The transient command files cannot be executed as long as they are in the same drive with other transient command files that have both the same primary file name and higher execution priorities. If you wish to execute such commands with lower priorities, use the RENAME command to give them a unique primary name or use the COPY command to copy them to a disk or partition in a different drive.

NOTE: If the entered command line function does not match a resident command or a transient command in the default or specified drive, then execution will not occur and the following error message will be displayed:

```
Bad command or file name
```

The system prompt would be displayed beneath this error message.

Supplementing the MSDOS.SYS Component

You can enhance your MS-DOS operating environment to accommodate your own preferences by creating or changing a CONFIG.SYS file. The system will automatically load a file by this name at bootup if it exists in the root directory of the booted disk or partition. The automatic loading of this file enables you to make additions or modifications to MSDOS.SYS that would otherwise take a great deal of time and experience to implement.

This file is an ASCII file containing subcommands for the completion of the boot sequence and configuration of certain system characteristics.

CONFIG.SYS Commands

The following commands can be included in a CONFIG.SYS file:

BUFFERS = *number*

In this command, *number* is the number of additional sector buffers to add to the system list. Initially the system list contains 2 disk buffers. Each occurrence of a BUFFERS command adds n buffers to the list. A buffer is about 528 bytes and can increase MS-DOS processing speed.

FILES = *number*

In this command, *number* is the number of open files that may be accessed by certain system calls. The default value is 8. If a value less than or equal to 5 is specified, the command is ignored.

DEVICE = *filename*

This command installs the device driver contained in *filename* into the system list. The *filename* specified must be a valid device driver. Your MS-DOS software includes the ANSI-compatible character driver ANSI.DVD. You can inplement this device driver with a DEVICE command in a CONFIG.SYS file.

BREAK = ON
BREAK = OFF

If ON is specified, a check for the CTRL-BREAK or CTRL-C entry at the console input is made every time the system is called. The default parameter for this command is OFF.

SWITCHAR = *char*

This command specifies the switch designation character. MS-DOS will return *char* as the current switch character when the operating system call to return the switch character is made. The default switch character is the forward slash mark (/).

NOTE: The switch character specified may affect characters used in the SHELL command. This is true for the default shell, COMMAND.COM.

AVAILDEV = TRUE
AVAILDEV = FALSE

The default for this command is TRUE. This means that *\dev\dev* and *dev* both reference the same device (that is, *dev*). If FALSE is selected, only *\dev\dev* refers to a device named *dev*. In this case, *dev* alone means a file in the current directory with the same name as one of the devices.

SHELL = *filename*

This command specifies the *filename* to be called to begin execution of the shell. The shell is the top level command processor, often referred to as the command interpreter. The default shell is COMMAND.COM, which is described in the section, Physical Components of this chapter.

CONFIG.SYS Example

A CONFIG.SYS file could contain the following command lines:

```
BUFFERS = 15
FILES = 10
DEVICE = \DEV\ANSI.SYS
BREAK = ON
```

This example supplements MSDOS.SYS with the following characteristics:

- Allocate 15 disk buffers,
- Allow 10 file channels,
- Load the ANSI.SYS device from the \dev directory, and
- Set BREAK to ON.

CONFIG.SYS Error Message

If a CONFIG.SYS file contains a command that is not in the form of the commands explained in this section, it will cause the following error message to be displayed when it is loaded during bootup:

```
Unrecognized command in CONFIG.SYS
```

Open the CONFIG.SYS file, correct its command(s) by using the described CONFIG.SYS command form(s), reset the system, and reboot.

Summary

Most microcomputer operating systems have the following five main generic components:

- Input/Output Handler – managing system input and output,
- File Handler – managing the file system,
- Memory Handler – managing allocation of microprocessor memory,
- Executive – loading programs and causing the microcomputer to execute the instructions in the program, and
- Boot Loader – beginning the process of loading the other system components (inactive after all components have been loaded).

This operating system performs all of the activities of the generic components, but is divided into these four MS-DOS components:

- MSDOS.SYS – hardware-independent director of activities,
- IO.SYS – hardware-dependent communicator with devices,

Summary

- COMMAND.COM – hardware-independent command interpreter, and
- MS-DOS Boot Loader – hardware-dependent mover of system from disk to memory.

The IO.SYS and MSDOS.SYS components can be recorded on disk media as hidden files by FORMAT and SYS. COMMAND.COM can be recorded by FORMAT, but not by SYS. COMMAND.COM is not a hidden file. The Boot Loader component can be recorded on disk media on a reserved nonfile sector by FORMAT.

When you boot up MS-DOS from storage media, its components perform several steps. The monitor program in Read Only Memory (ROM) loads the boot loader from the media (if the media is a floppy disk) or it loads the boot record (if the media is a Winchester partition). (If the media is a Winchester partition, the boot record loads the boot loader.) The boot loader then loads the files IO.SYS, and IO.SYS loads MSDOS.SYS. SYSINIT (a module of IO.SYS) relocates and initializes MSDOS.SYS. Then SYSINIT loads CONFIG.SYS (if present) and COMMAND.COM (unless another shell is specified). COMMAND.COM loads AUTOEXEC.BAT (if present). After execution of either of the commands in AUTOEXEC.BAT or the DATE and TIME commands, COMMAND.COM displays the system prompt.

It is possible for the function of an entered command to apply to more than one accessible command. When this happens, COMMAND.COM determines that the command to be executed is the highest available command in the following list of execution priorities:

- Resident command (resides in memory and has no file name extension)
- Transient command with the .COM file name extension
- Transient command with the .EXE file name extension
- Transient command with the .BAT file name extension

You can customize the MSDOS.SYS component by creating or changing an ASCII file named CONFIG.SYS and then rebooting.

Part III

Primary Command Guide

Primary Command Guide

This guide provides a comprehensive description of the commands in your MS-DOS software package that will be beneficial to the average user.

Knowledge of all these commands is not essential for most users who only wish to run an application program under MS-DOS. However, knowledge of many of these commands can make MS-DOS a more powerful, convenient, and versatile software tool.

A *primary command* is a program designed to accomplish a goal that might be common to all users of MS-DOS with a microcomputer.

These commands are explained in terms that can be understood by users with no prior microcomputer experience. To use this guide, you should understand all of the concepts and perform all of the necessary procedures in Part I, "Preparation Guide" of this manual.

Familiarity with some sections of Part II, "Primary Feature Guide" is also helpful for using some of these commands.

The Primary Command Guide consists of the following two chapters:

Chapter 10, "Command Summaries"—This chapter includes two kinds of summaries of the primary commands: alphabetic and functional. Each summary lists the name, purpose, and entry form(s) of the primary commands.

The alphabetic summary presents each primary command once, arranged in alphabetic order by command name. The functional summary presents every primary command at least once, arranged in categories according to the function(s) of the command. Some commands will appear in more than one functional category if they have more than one function.

Chapter 11, "Command Descriptions"—This chapter describes the primary commands provided with your MS-DOS software.

The command descriptions are arranged in alphabetic order according to command name. Most command sections include the following items:

- Purpose – Brief paragraph explaining what the command does.

- Entry Form(s) – Demonstration of how the command should be entered, consisting of mandatory entries (entered verbatim each time the command is used) and variable entries (entered differently in each situation.)

- Preliminary Concepts – Explanation of some hardware or software aspect you should know before using the command.

- Examples – Demonstrations of how the command can be used to achieve a specific goal. The titles of the example sub-sections usually identify the intended goal.

- Error Messages – Display of each error message, explanation of why it might have occurred, and suggestions for recovering from the error (if possible).

NOTE: These commands conform to the definition of commands found in Chapter 5, "Command Features" of Part II, "Primary Feature Guide," and they can be entered according to the command entry rules explained in Chapter 5.

Other commands are also included with your MS-DOS software. However, these other commands are intended for users with special hardware and/or software needs.

If you need information about commands that help with writing programs, refer to Part IV, "Program Development Command Guide."

If you need information about commands for preparing a Winchester disk, refer to Part V, "Winchester Command Guide."

Chapter 10

Command Summaries

Overview

This chapter summarizes some of the information found in Chapter 11, "Command Descriptions" emphasizing the formats in which MS-DOS primary commands operate, and the special groupings that apply to the commands. The commands are presented both alphabetically and functionally.

Some commands contain lowercase italic variables that represent the following:

afn	Ambiguous file name
***d*:**	Drive name or destination drive name
p	Winchester disk partition
***s*:**	Source drive name
***u*:**	Unit number
/x	Switch

Alphabetical Summary

A complete alphabetical listing of MS-DOS commands by name, purpose, and command line entry form comprises Table 10.1. This is a quick reference arranged in the same sequence as the commands listed in Chapter 11, "Command Descriptions."

Each command name is followed by an R (for resident) or a T (for transient). Resident commands are located in the operating system itself. Transient commands are located in an executable file that is recorded on disk.

Command Summaries

Alphabetical Summary

Table 10.1. Alphabetical Summary of Command Descriptions

COMMAND	R/T	PURPOSE	ENTRY FORM
APPLY	T	Executes a command with substitution.	APPLY [*d*:]*filename* "*command*" APPLY "*command*" [*d*:]*filename* APPLY [*d*:]*pathname* "*command*" APPLY "*command*" [*d*:]*pathname* APPLY [-] "*command*" APPLY "*command*" [-]
ASSIGN	T	Assigns a Winchester partition to a logical drive letter.	ASSIGN[?] ASSIGN u: ASSIGN *u*:*p* *d*:
BACKUP	T	File archiver, creates a backup copy of partition or disk files.	BACKUP BACKUP ? BACKUP [*filespec*[+*filespec*...]] [*d*:] [*filename*][/*x*...]
BREAK	R	Checks for CTRL-BREAK or CTRL-C. BREAK ON checks for all occurrences.	BREAK [ON] BREAK [OFF]
CHDIR (CD)	R	Displays or changes the current directory.	CHDIR [*d*:][*pathname*] CD [*d*:][*pathname*]
CHKDSK	T	Provides the status of a disk's contents.	CHKDSK [*d*:][*filename*][/*x*]
CIPHER	T	Encrypt and decrypt files.	CIPHER *keyword* >*filespec* CIPHER *keyword* <*filespec* CIPHER *keyword* <*filspec1* >*filspec2*
CLS	R	Clears the screen.	CLS
COMMAND	T	Makes EXEC calls on resident commands.	COMMAND [*d*:][*pathname*][*cttydev*][/*x*]
COMP	T	Compares files	COMP? COMP [*filspec1* [*filspec2*]]
CONFIGUR	T	Configures MS-DOS for your specific hardware.	CONFIGUR

Table 10.1 (continued). Aphabetical Summary of Command Descriptions

COMMAND	R/T	PURPOSE	ENTRY FORM
COPY	R	Copies file or files specified.	COPY *filespec d:* [/V] COPY *filespec* [*d:*]*filename*[/V] COPY *filespec* [*d:*]*pathname*[/V] COPY [*d:*]*pathname* [*d:*][*pathname*][/V]
CTTY	R	Changes the device from which commands are issued.	CTTY *device*
DATE	R	Displays and sets the date.	DATE DATE *mm-dd-yy*
DEL (ERASE)	R	Deletes file or files specified.	DEL *filespec* DEL [*d:*]*pathname*\filename ERASE *filespec* ERASE [*d:*]*pathname\filename*
DIR	R	Lists requested directory entries.	DIR [*d:*][*filename*][/*x*] DIR [*d:*][*pathname*][/*x*]
DISKCOMP	T	Compares disks.	DISKCOMP [*s:* [*d:*]]
DISKCOPY	T	Copies disks.	DISKCOPY[*s:* [*d:*]][/V]
ECHO	R	Controls the echo feature.	ECHO [ON] ECHO [OFF] ECHO [*message*]
ERASE	R	See DEL command.	
EXIT	R	Exits command and returns to lower level.	EXIT
FC	T	Lists differences between files specified.	FC *filename1 filename2*[/*x*]
FIND	T	Searches for a constant string of text.	FIND "*string*" [*filespec...*][/*x*]

Table 10.1 (continued). Alphabetical Summary of Command Descriptions

COMMAND	R/T	PURPOSE	ENTRY FORM
FOR	R	Batch and interactive command extension.	FOR %*variable* IN (*set*) DO *command* FOR %%*variable* IN (*set*) DO *command*
FORMAT	T	Formats a disk to receive MS-DOS files.	FORMAT [*d*:] [/*x*...]
GOTO	R	Branches out of execution.	GOTO *label*
IF	R	Allows conditional execution.	IF[NOT] *condition command*
MAP	T	Temporarily reassigns logical drives names	MAP? MAP [*x*=*y*[m]]
MKDIR (MD)	R	Makes a new directory.	MKDIR [*d*:]*pathname* MD [*d*:]*pathname*
MODE	T	Configures MS-DOS for your specific hardware.	MODE ? MODE LPT#: [*n*] [, [*m*] [,P]] MODE [*n*] [, [*m*] [, [T] [, *s*]] MODE COM*n*: *baud*[, [*parity*] [, [*databits*] [, [*stopbits*] [,P]]]] MODE LPT#: =COM*n*
MORE	T	Displays output one screen at a time.	MORE *command* | MORE
PATH	R	Specifies directories to be searched.	PATH [*d*:] [*pathname*[; [*d*:]*pathname*]...]
PAUSE	R	Suspends execution.	PAUSE [*remark*]
PRINT	T	Prints hard copy of ASCII files.	PRINT PRINT *filespec*[/*x*] [*filespec*[/*x*]...]
PROMPT	R	Designates MS-DOS system prompt.	PROMPT [$] [*prompttext*]

Table 10.1 (continued). Alphabetical Summary of Command Descriptions

COMMAND	R/T	PURPOSE	ENTRY FORM
PSC *printer*	T	Output all graphic and special characters to printer.	PSC*printername*
RDCPM	T	Copies many CP/M files.	RDCPM RDCPM ? RDCPM DIR *d*: [*filename*] [/Z] RDCPM [*s*:] *sorcfile* [*d*:] [*destfile*] [/Z]
RECOVER	T	Recovers file or files specified.	RECOVER *d*: RECOVER *filespec*
REM	R	Displays a batch file comment.	REM [*remark*] . [*remark*]
REN (RENAME)	R	Renames first file as second file name.	REN *filespec filename* RENAME *filespec filename*
RESTORE	T	Restores archived files.	RESTORE RESTORE ? RESTORE [[*d*:]*filename* [*filespec* [+*filespec*...]]] [/*x*...]
RMDIR (RD)	R	Removes a directory.	RMDIR [*d*:]*pathname* RD [*d*:]*pathname*
SEARCH	T	Locates files within directory structure.	SEARCH[*afn*] [/*x*]
SET	R	Sets one string value equivalent to another.	SET SET [*string1*=*string2*]
SHIFT	R	Allows use of over 10 batch parameters.	SHIFT
SORT	T	Sorts data alphabetically or numerically.	SORT [/*x*]
SYS	T	Transfers system files IO.SYS and MSDOS.SYS to specified drive.	SYS *d*:

Table 10.1 (continued). Alphabetical Summary of Command Descriptions

COMMAND	R/T	PURPOSE	ENTRY FORM
TIME	R	Displays and sets the time.	TIME TIME *hh*[:*mm*[:*ss*[.*cc*]]]
TREE	T	Displays subdirectory paths on a disk.	TREE ? TREE [*d*:][/F]
TYPE	R	Displays the contents of the ASCII file specified.	TYPE *filespec* TYPE [*d*:][*pathname*]*filename*
VER	R	Displays MSDOS.SYS and IO.SYS version numbers.	VER
VERIFY	R	Verifies data is correctly written to disk.	VERIFY [ON] VERIFY [OFF]
VOL	R	Displays disk volume label.	VOL [*d*:]

Functional Summary

This section contains a complete functional listing of MS-DOS commands by name, purpose, and command line entry form. This is a quick reference, by function, for the commands listed in Chapter 11, "Command Descriptions."

The functional summary presents every primary command at least once, arranged in categories according to the function of the command. Some commands will appear in more than one category.

Each command's name is followed by an R (for resident) or a T (for transient). Resident commands are located in the operating system itself. Transient commands are located in executable files which are recorded on disk.

Batch Processing Commands

The following commands control the display and execution of MS-DOS commands during the execution of a batch file. These commands can be executed from within a batch file or directly at a system prompt.

COMMAND	R/T	PURPOSE	ENTRY FORM

ECHO	R	Controls the echo feature.	`ECHO [ON]`
			`ECHO [OFF]`
			`ECHO [`*`message`*`]`

FOR	R	Batch and interactive command extension.	`FOR %`*`variable`*` IN (`*`set`*`) DO `*`command`* `FOR %%`*`variable`*` IN (`*`set`*`) DO `*`command`*
GOTO	R	Branches out of execution.	`GOTO `*`label`*
IF	R	Allows conditional execution.	`IF[ NOT] `*`condition command`*
PAUSE	R	Suspends execution.	`PAUSE [`*`remark`*`]`
REM	R	Displays a batch file comment.	`REM [`*`remark`*`]` `. [`*`remark`*`]`
SET	R	Sets one string value equivalent to another.	`SET` `SET [`*`string1=string2`*`]`
SHIFT	R	Allows use of over 10 batch parameters.	`SHIFT`

Command Processor

This command interprets commands, directs functions, and makes sure the correct programs are executed.

COMMAND	R/T	PURPOSE	ENTRY FORM
COMMAND	T	Makes EXEC calls on resident commands.	`COMMAND [d:] [pathname] [cttydev] [/x]`

Convenience Commands

The following commands are used to make general MS-DOS operation convenient, but are not essential to most MS-DOS operations.

COMMAND	R/T	PURPOSE	ENTRY FORM
CLS	R	Clears the screen.	`CLS`
DATE	R	Displays and sets the date.	`DATE` `DATE mm-dd-yy`
EXIT	R	Exits command and returns to lower level.	`EXIT`
MAP	T	Temporarily reassigns logical drive names.	`MAP ?` `MAP [x=y [ . . . ]]`
PROMPT	R	Designates MS-DOS system prompt.	`PROMPT [$] [prompttext]`
REM	R	Displays a batch file comment.	`REM [remark]` `. [remark]`
SET	R	Sets one string value equivalent to another.	`SET` `SET [string1=string2]`

COMMAND	R/T	PURPOSE	ENTRY FORM
TIME	R	Displays and sets the time.	`TIME` `TIME hh[:mm[:ss[.cc]]]`
TREE	T	Displays subdirectory paths on a disk.	`TREE ?` `TREE [d:][/F]`
VER	R	Displays MSDOS.SYS and IO.SYS version numbers.	`VER`
VOL	R	Displays disk volume label.	`VOL [d:]`

Data Filtering Commands

The following commands are filter commands that read from standard input (such as command input) and manipulate data. These commands are easily used with pipes to afford greater flexibility in application.

COMMAND	R/T	PURPOSE	ENTRY FORM

CIPHER	T	Encrypt and decrypt files.	`CIPHER keyword >filespec`
			`CIPHER keyword <filespec`
			`CIPHER keyword <filspec1 >filspec2`

FIND	T	Searches for a constant string of text.	`FIND "string" [filespec...][/x]`
MORE	T	Displays output one screen at a time.	`MORE` `command | MORE`
SORT	T	Sorts data alphabetically or numerically.	`SORT [/x]`

Data/Software Analysis

The following commands are used to separate, examine, and catalog your disks and files.

COMMAND	R/T	PURPOSE	ENTRY FORM
BACKUP	T	Displays contents of a backup file or backup disk.	BACKUP [*d*:]/D BACKUP [*d*:]*filespec*/L
CHDIR (CD)	R	Displays the current directory.	CHDIR [*d*:] CD [*d*:]
CHKDSK	T	Provides the status of a disk's contents.	CHKDSK [*d*:] [*filename*] [/V]
COMP	T	Compares files.	COMP ? COMP [*filespc1* [*filespc2*]]
DATE	R	Displays and sets the date.	DATE DATE *mm-dd-yy*
DIR	R	Lists requested directory entries.	DIR [*d*:] [*filename*] [/*x*] DIR [*d*:] [*pathname*] [/*x*]
DISKCOMP	T	Compares disks.	DISKCOMP [*s*: [*d*:]]
FC	T	Lists differences between files specified.	FC *filename1 filename2* [/*x*]
RESTORE	T	Restores archived files.	RESTORE RESTORE ? RESTORE [[*d*:]*filename* [*filespec*[+ *filespec*...]]] [/*x*...]
TIME	R	Displays and sets the time.	TIME TIME *hh*[:*mm*[:*ss*[.*cc*]]]
TREE	T	Displays subdirectory paths on a disk.	TREE ? TREE [*d*:] [/F]

COMMAND	R/T	PURPOSE	ENTRY FORM
TYPE	R	Displays the contents of the ASCII file specified.	`TYPE filespec` `TYPE [d:][pathname]filename`
VER	R	Displays MSDOS.SYS and IO.SYS version numbers.	`VER`
VOL	R	Displays disk volume label.	`VOL [d:]`

Data/Software Change

The following commands are used to create a change in the status of your disks and/or files.

COMMAND	R/T	PURPOSE	ENTRY FORM

ASSIGN	T	Assigns a Winchester partition to a logical drive letter.	`ASSIGN[ ?]`
			`ASSIGN u:`
			`ASSIGN u:p d:`

CHDIR (CD)	R	Displays or changes the current directory.	`CHDIR [d:][pathname]` `CD [d:][pathname]`
CHKDSK	T	Provides the status of a disks contents.	`CHKDSK [d:][filename][/x]`

DEL
(ERASE)
	R	Deletes file or files specified.	`DEL filespec`
			`DEL [d:]pathname\filename`
			`ERASE filespec`
			`ERASE [d:]pathname\filename`

FORMAT	T	Formats a disk to receive MS-DOS files.	`FORMAT [d:][/x...]`
MAP	T	Temporarily reassigns logical drive names.	`MAP ?` `MAP [x=y [ . . . ]]`

COMMAND	R/T	PURPOSE	ENTRY FORM
PATH	R	Specifies directories to be searched.	PATH [*d:*][*pathname*[;[*d:*]*pathname*]...]
REN (RENAME)	R	Renames first file as second file name.	REN *filespec filename* RENAME *filespec filename*
RMDIR (RD)	R	Removes a directory.	RMDIR [*d:*]*pathname* RD [*d:*]*pathname*

Data/Software Creation

The following commands are used to create files, directories, or disks.

COMMAND	R/T	PURPOSE	ENTRY FORM
BACKUP	T	File archiver, creates a backup copy of partition or disk files.	BACKUP BACKUP ? BACKUP [*filespec*[+*filespec*...]] [*d:*] [*filename*][/*x*...]
COPY	R	Creates a file using input from a device.	COPY *filespec* [*d:*]*filename*[/V]
DISKCOPY	T	Copies disks.	DISKCOPY[*s:* [*d:*]][/V]
FORMAT	T	Copies system components to disk media.	FORMAT [*d:*][/*x*...]
MKDIR (MD)	R	Makes a new directory.	MKDIR [*d:*]*pathname* MD [*d:*]*pathname*

COMMAND	R/T	PURPOSE	ENTRY FORM

RDCPM	T	Copies many CP/M files.	`RDCPM`
			`RDCPM ?`
			`RDCPM DIR d: [filename][/Z]`
			`RDCPM [s:] sorcfile [d:][destfile][/Z]`

RESTORE	T	Restores archived files.	`RESTORE`
			`RESTORE ?`
			`RESTORE [[d:]filename [filespec[+filespec...]]][/x...]`

SYS	T	Transfers system files IO.SYS and MSDOS.SYS to specified drive.	`SYS d:`

Data/Software Movement

The following commands are used to move software or data (usually files) between devices (usually disk drives).

COMMAND	R/T	PURPOSE	ENTRY FORM

BACKUP	T	File archiver, creates a backup copy of a partition or disk files.	`BACKUP`
			`BACKUP ?`
			`BACKUP [filespec[+filespec...]] [d:][filename][/x...]`

COPY	R	Copies file or files specified.	`COPY filespec d:[/V]`
			`COPY filespec [d:]filename[/V]`
			`COPY filespec [d:]pathname[/V]`
			`COPY [d:]pathname [d:][pathname][/V]`

DISKCOPY	T	Copies disks.	`DISKCOPY[ s: [d:]][/V]`

COMMAND	R/T	PURPOSE	ENTRY FORM
FORMAT	T	Copies system components to disk media.	FORMAT [*d*:][/x...]
RECOVER	T	Recovers file or files specified.	RECOVER *d*: RECOVER *filespec*
RESTORE	T	Restores archived files.	RESTORE RESTORE ? RESTORE [[*d*:]*filename* [*filespec* [+*filespec*...]]][/x...]
SYS	T	Transfers system files IO.SYS and MSDOS.SYS to specified drive.	SYS *d*:

Directory Commands

The following commands are used to create and control the directory structure of MS-DOS version 2. These commands allow you to access files through multiple levels of directories.

COMMAND	R/T	PURPOSE	ENTRY FORM
CHDIR (CD)	R	Displays or changes the current directory.	CHDIR [*d*:][*pathname*] CD [*d*:][*pathname*]
MKDIR (MD)	R	Makes a new directory.	MKDIR [*d*:]*pathname* MD [*d*:]*pathname*
PATH	R	Specifies directories to be searched.	PATH [*d*:][*pathname*[;[*d*:]*pathname*]...]
RMDIR (RD)	R	Removes a directory.	RMDIR [*d*:]*pathname* RD [*d*:]*pathname*
SEARCH	T	Locates files within directory structure.	SEARCH [*afn*][/x]

Hard Copy Commands

The two following commands are used to send data from an ASCII file or the screen to the peripheral device that is configured as the PRN device.

COMMAND	R/T	PURPOSE	ENTRY FORM
PRINT	T	Prints hard copy of ASCII files.	`PRINT` `PRINT filespec[/x] [filespec[/x]...]`
PSC *printer*	T	Output all graphics and special characters to printer.	`PSCprintername`

Manipulation Commands

The following commands are used to change, duplicate, delete, or recover files.

COMMAND	R/T	PURPOSE	ENTRY FORM

APPLY	T	Executes a command with substitution.	`APPLY [d:]filename "command"`
			`APPLY "command" [d:]filename`
			`APPLY [d:]pathname "command"`
			`APPLY "command" [d:]pathname`
			`APPLY [-] "command"`
			`APPLY "command" [-]`

BACKUP	T	File archiver, creates a backup copy of partition or disk files.	`BACKUP`
			`BACKUP ?`
			`BACKUP [filespec[+filespec...]] [d:] [filename][/x...]`

Command Summaries

Functional Summary

COMMAND	R/T	PURPOSE	ENTRY FORM
CIPHER	T	Encrypts and decrypts files.	CIPHER *keyword* >*filespec* CIPHER *keyword* <*filespec* CIPHER *keyword* <*filspec1* >*filspec2*
COPY	R	Copies file or files specified.	COPY *filespec d*: [/V] COPY *filespec* [*d*:]*filename*[/V] COPY *filespec* [*d*:]*pathname*[/V] COPY [*d*:]*pathname* [*d*:][*pathname*][/V]
DEL (ERASE)	R	Deletes file or files specified.	DEL *filespec* DEL [*d*:]*pathname*\filename ERASE *filespec* ERASE [*d*:]*pathname\filename*
DISKCOPY	T	Copies disks.	DISKCOPY [*s*: [*d*:]][/V]
RDCPM	T	Copies many CP/M files.	RDCPM RDCPM ? RDCPM DIR *d*: [*filename*][/Z] RDCPM [*s*:] *sorcfile* [*d*:][*destfile*][/Z]
RECOVER	T	Recovers file or files specified.	RECOVER *d*: RECOVER *filespec*
REN (RENAME)	R	Renames first file as second file name.	REN *filespec filename* RENAME *filespec filename*
RESTORE	T	Restores archived files.	RESTORE RESTORE ? RESTORE [[*d*:]*filename* [*filespec* [+*filespec*...]]][/*x*...]

Safety Commands

The following commands are used to help you prevent, or recover from, some unexpected problems in MS-DOS operations.

COMMAND	R/T	PURPOSE	ENTRY FORM
BREAK	R	Checks for CTRL-BREAK or CTRL-C. BREAK ON checks for all occurrences.	BREAK [ON] BREAK [OFF]
CHKDSK	T	Provides the status of a disk's contents.	CHKDSK [*d*:] [*filename*] [/*x*]
COMP	T	Compares files.	COMP ? COMP [*filespc1* [*filespc2*]]
DISKCOMP	T	Compares disks.	DISKCOMP [*s*: [*d*:]]
FC	T	Lists differences between files specified.	FC *filename1 filename2*[/*x*]
RECOVER	T	Recovers file or files specified.	RECOVER *d*: RECOVER *filespec*
VERIFY	R	Verifies data is correctly written to disk.	VERIFY [ON] VERIFY [OFF]

System Preparation Commands

The following commands are used to prepare MS-DOS software for use in your hardware environment.

COMMAND	R/T	PURPOSE	ENTRY FORM
CONFIGUR	T	Configures MS-DOS for your specific hardware.	CONFIGUR
CTTY	R	Changes the device from which commands are issued.	CTTY *device*
FORMAT	T	Formats a disk to receive MS-DOS files.	FORMAT [*d*:] [/*x*. . .]
MODE	T	Configures MS-DOS for your specific hardware.	MODE ? MODE LPT#: [*n*] [, [*m*] [,P]] MODE [*n*] [,d[*m*] [, [T] [,*s*]] MODE COM*n*:*baud*[, [*parity*] [, [*databits*] [, [*stopbits*] [,P]]]] MODE LPT#: =COM*n*
SET	R	Sets one string value equivalent to another.	SET SET [*string1*=*string2*]
SYS	T	Transfers system files IO.SYS and MSDOS.SYS to specified drive.	SYS *d*:

Chapter 11
Command Descriptions

This chapter provides a comprehensive description of each primary command provided with your MS-DOS software. The command descriptions are arranged in alphabetic order according to command name. Most command sections include the purpose, entry form(s), preliminary concepts, examples, and error messages of the command.

NOTE: Refer to Chapter 5, "Command Features," of Part II, "Primary Feature Guide" for more information on command entry. Refer to the "Introduction" of the manual for a comprehensive list of definitions for the variables used in the entry forms of this chapter.

ANSI.SYS Terminal Driver

Purpose

ANSI.SYS is one of two functional examples of user-loadable device drivers that are provided on your MS-DOS version 2 distribution disks. The other device driver is MDISK.DVD. While ANSI and MDISK are *not* commands or utilities that are invoked at the system prompt, they are described in this chapter because they may be loaded and used by user option.

ANSI.SYS is a functional character device driver that enables you to use special character sequences (escape codes) in your programs to control cursor positioning in screen displays. The special character sequences also enable you to redefine the function or meaning of any key in the keyboard. Essentially, ANSI.SYS provides ANSI terminal emulation.

The ANSI driver can be added to the system's linked list of standard device drivers (COM1:, LPT1:, and so on, as described in Chapter 8, "Input/Output Features") by inserting a DEVICE= command in the CONFIG.SYS file in the root directory. Note that this is the *only* way that ANSI can be installed and used; it is loaded into memory at bootup only if it is specified in the CONFIG.SYS file. For more information about the CONFIG.SYS file, refer to Chapter 9, "System Component Features."

Installing ANSI.SYS

To load (install) the ANSI driver, you must include a command line in the following form in a CONFIG.SYS file in the root directory:

```
DEVICE=ANSI.SYS
```

Once you have done this, ANSI.SYS will be loaded into memory whenever you boot up your system.

Using ANSI

During system operation, the ANSI device will respond just as any other device that is included in the system list. ANSI is basically a CON type device, in that it drives the keyboard and monitor. The difference between ANSI and CON is that ANSI translates ANSI escape codes into special console functions.

The special character sequences supported by ANSI are valid *only* when issued through MS-DOS function calls 1, 2, 6, and 9. Supported character sequences provide for cursor control, erasing, the definition of operational modes, and the redefinition of keys. The escape codes for supported functions are provided in the listings that follow.

For all escape codes listed, the letters "ESC" represent the 1-byte code for ESC (that is, hex 1B), *not* alpha characters. Additionally, the # character is used to designate a decimal number specified with ASCII characters. A default value is used when no explicit value is used and/or when a value of zero is specified.

Cursor Control

FUNCTION	MNEMONIC	ESCAPE CODE	DESCRIPTION
Cursor Position	CUP	ESC[#;#H	Moves the cursor to the position specified by the numerical parameters. The first parameter specifies the line number; the second parameter specifies the column. The default for both the line and column numbers is 1. If no parameters are specified, the cursor is moved to the home position.
Cursor Up	CUU	ESC [#A	Moves the cursor up in the same column. The value of # specifies the number of lines moved. The default is 1 line. This escape code is ignored if the cursor is already on the top line of the display.
Cursor Down	CUD	ESC [#B	Moves the cursor down in the same column. The value of # specifies the number of lines moved. The default is 1 line. This escape code is ignored if the cursor is already on the bottom line of the display.

FUNCTION	MNEMONIC	ESCAPE CODE	DESCRIPTION
Cursor Forward	CUF	ESC[#C	Moves the cursor forward (right) in the same line. The value of # specifies the number of columns moved. The default is 1 column. This escape code is ignored if the cursor is already in the rightmost column.
Cursor Back	CUB	ESC[#D	Moves the cursor back (left) in the same line. The value of # specifies the number of columns moved. The default is 1 column. This escape code is ignored if the cursor is already in the leftmost column.
Horizontal and Vertical Position	HVP	ESC[#;#f	Moves the cursor to the position specified by the numerical parameters. The first parameter specifies the line number and the second parameter specifies the column number. The default value for both parameters is 1. If no parameters are specified, the cursor is moved to the home position (same as CUP).

ANSI.SYS

FUNCTION	MNEMONIC	ESCAPE CODE	DESCRIPTION
Device Status Report	DSR	ESC[6n	Causes console driver to output a CPR sequence. (CPR sequence, as described below, is output on receipt of DSR sequence.)
Cursor Position Report	CPR	ESC[#;#R	This sequence reports the current cursor position through the standard input device. The first numerical parameter specifies the current line and the second parameter specifies the current column.
Save Cursor Position	SCP	ESC[s	Saves the current cursor position. The cursor position saved with this sequence can be restored with the RCP sequence.
Restore Cursor Position	RCP	ESC[u	Restores the cursor to the position (value) it had when the console driver received the SCP sequence.

Erasing

FUNCTION	MNEMONIC	ESCAPE CODE	DESCRIPTION
Erase in Display	ED	ESC[2J	Clears (erases) the entire display and returns the cursor to the home position.
Erase in Line	EL	ESC[K	Erases from the cursor (including the cursor position) to the end of the line.

Mode of Operation

FUNCTION	MNEMONIC	ESCAPE CODE	DESCRIPTION
Set Graphics Rendition	SGR	ESC[#;...;#m	Sets the character attribute(s) specified by the numerical parameter(s). All subsequent characters, until the next SGR sequence, will have the specified attributes. The valid numerical parameters and their meanings are listed below: 0 All attributes off (normal black on white) 1 Bold on (high intensity) 4 Underscore on (monochrome display only) 5 Blink on 7 Reverse video on 8 Cancelled on (invisible) 30 Black foreground 31 Red foreground 32 Green foreground 33 Yellow foreground 34 Blue foreground 35 Magenta foreground 36 Cyan foreground 37 White foreground 40 Black background 41 Red background 42 Green background 43 Yellow background 44 Blue background 45 Magenta background 46 Cyan background 47 White background

Set Mode	SM	ESC[=#h ESC[=h ESC[=0h ESC[?7h	Invokes the screen width or type specified by the numerical parameter. The valid numerical parameters and their meanings are listed below: 0 40 × 25 black and white 1 40 × 25 color 2 80 × 25 black and white 3 80 × 25 color 4 320 × 200 color 5 320 × 200 black and white 6 640 × 200 black and white 7 Line wrap on (typing past end of line creates new line)
FUNCTION	MNEMONIC	ESCAPE CODE	DESCRIPTION
Reset	RM	ESC[=#l ESC[=l ESC[=0l ESC[?7l	Resets the screen width or type specified by the numerical parameter. The valid numerical parameters and their meanings are the same as for the SM sequence, except that 7 turns line wrap off (characters typed past the end of a line are discarded).

Reassignment of Keyboard Key Functions

ESCAPE CODE	DESCRIPTION
ESC[#;#;...#p ESC["*string*";p ESC[#;"*string*";#;#;"*string*";#p	The first numerical parameter (ASCII code) in the sequence defines the code being redefined or remapped. The remaining numerical parameters define the sequence of ASCII codes generated when the key is intercepted. Note, however, that if the first code in the sequence is zero (NUL), then the first and second code make up an extended ASCII redefinition.

NOTE: In addition to the escape code sequences shown, the sequence may be any combination of strings and decimal numbers.

For example, to reassign the "Q" and "q" key to be the "A" and "a" key, respectively, the following escape codes would be used:

- `ESC[65;81p` (A becomes Q)
- `ESC[97;113p` (a becomes q)
- `ESC[81;65p` (Q becomes A)
- `ESC[113;97p` (q becomes a)

To redefine function key F10 to be a CHDIR command followed by a RETURN, you would use the following escape code:

```
ESC[0;68;"chdir";13p
```

In this example, 0;68 is the extended ASCII code for function key F10. Decimal 13 is a RETURN. If F10 is redefined in this way, the current working directory of the default disk will be displayed each time F10 is pressed.

APPLY (Transient)

Purpose

Executes a given command multiple times, with substitution of a selected parameter.

NOTE: To successfully use this command, a valid COMMAND.COM file must be on the disk in the default drive.

Entry Forms

```
APPLY [d:]filename "command"
APPLY [d:]pathname "command"
APPLY [-] "command"
APPLY "command" [d:]filename
APPLY "command" [d:]pathname
APPLY "command" [-]
```

where ***d***: is the drive name identifying the drive in which the source file (that is, the file to be input to APPLY for parameter substitution) is located;
filename is the file name of the file you want to input to APPLY;

pathname is the directory path name (including a file name) you want to input to APPLY;
- is the standard input; and
command is the command you want to execute, using as parameters the input to APPLY.

NOTE: The MS-DOS standard input is the default source for APPLY.

Preliminary Concepts

The APPLY command enables you to execute a specified command more than once by using parameters or input from the specified source. The default source is the standard input. This command may be used with pipes to receive input from another MS-DOS command or function. (Refer to Chapter 8, "Input/Output Features," for more information about the standard input and pipes.)

When you specify a file as input to APPLY, APPLY reads lines from the specified file, substitutes the lines for a parameter in the command you specify, and executes the command once for each parameter substitution. In essence, APPLY causes the command you specify to be executed once for each line of input. If you do not specify a source, the standard input will be used as parameter(s) in the specified command.

During execution of the APPLY command, the name of the command being executed through APPLY and the parameter currently being used in the command are displayed. Refer to Examples in this section.

Command Line Entry

The command line parameters required in order to invoke the APPLY command are described below. Pressing the SPACE BAR or using an equivalent MS-DOS delimiter is required between the parameters entered as part of any one APPLY command line.

You must always specify the command you want applied to the default or specified source. However, the order in which you specify the source parameter and the command parameter does not matter, as long as they both follow the command name, APPLY.

Source Specification

If you do not want APPLY to read the standard input, you must specify the source to be read, and from which parameters will be substituted in the command you choose for execution. You may specify either of the following as the source of input to APPLY:

- [*d*:]***filename***, if you want a file in the current directory of the default or specified disk to be read as input.
- [*d*:]***pathname***, if you want a file in a directory other than the current directory of the default or specified disk to be read as input.

In either case, if the file is not on the default disk, you must specify the appropriate drive name (*d*:) as part of the source specification.

If you want APPLY to read the standard input, you may either specify no source parameter, or you may specify a hyphen (-) as the source parameter. That is, you may enter a command in the form

```
APPLY "command"
```

or

```
APPLY - "command"
```

and APPLY will read the standard input as its source. The default standard input is the keyboard; standard input may be output from another command when pipes and/or input/output redirection are used. Thus, the above entry forms may be used when input to APPLY is piped from another MS-DOS command. (Refer to Chapter 8, "Input/Output Features," for more information on pipes and input/output redirection.)

Like input from a file, the standard input is read line by line and each line is used once as a parameter in the command you are executing through APPLY.

Command

You must always specify the *command* to be executed using the input to APPLY; there is no default *command*. The command you specify *must* be enclosed in quotation marks, and must be a valid MS-DOS command. The command must follow the command line entry requirements that are applicable to it, with the condition that you may represent one parameter (or part of one parameter, as may be the case with file specifications) with the **%** character that designates a variable. Otherwise, there are no restrictions or requirements for the *command* parameter.

The *command* you specify will be executed once for each line of input to APPLY, where the line is substituted for a *command* parameter designated by %. That is, occurrences of **%** in the *command* will be replaced with one line at a time from the input to APPLY. The *command* will be executed after each parameter substitution.

Examples

Suppose you have the file ALLBAK.FIL on the disk in drive B and that this file consists of all the .BAK files on that disk. If you wished to copy all of the .BAK files to another disk for storage and then delete them from your working disk, you could perform the following:

- At the system prompt, enter

  ```
  APPLY B:ALLBAK.FIL "COPY B:% C:"
  ```

  and press **RETURN**.

- The system will begin copying all files listed in the file ALLBAK.FIL from the disk in drive B to the disk in drive C. Each time the COPY command is executed, the screen displays

  ```
  COPY B:filename.BAK C:
  ```

  where *filename* is the primary file name of the file currently being copied.

  After a given file is successfully copied, the screen displays

  ```
  1 File(s) copied
  ```

  before going on to the next file listed in ALLBAK.FIL.

- When all files listed in the file ALLBAK.FIL have been copied, the system prompt is displayed again.

- To delete the .BAK files from your working disk, enter

  ```
  APPLY B:ALLBAK.FIL "ERASE %"
  ```

  and press **RETURN**.

- The system will begin deleting all files listed in the file ALLBAK.FIL from the disk in drive B. Each time the ERASE command is executed, the screen displays

  ```
  ERASE B:filename.BAK
  ```

  where *filename* is the primary file name of the file being deleted.

- When all files listed in the file ALLBAK.FIL have been deleted, the system prompt is displayed again.

Error Messages

```
Error EXECing COMMAND.COM, Terminating
```

EXPLANATION: This message is displayed if you attempted to execute APPLY without having a valid COMMAND.COM file on the disk in the default drive. This file must be available on the default disk in order for APPLY to execute.

```
Unable to locate requested input file!
```

EXPLANATION: This message is displayed if you enter an invalid file name or path name, or if the system cannot locate the source you entered on the specified disk. Reenter the APPLY command, making sure that the source is specified correctly and is available on the appropriate disk.

ASSIGN (Transient)

Purpose

The ASSIGN command is used to assign a physical Winchester drive partition number to a logical drive name. This makes a logical connection between a drive name and a valid Winchester drive partition so that data can be moved easily to and from these partitions.

Entry Form

```
ASSIGN[ ?]
ASSIGN u:
ASSIGN u:p d:
```

where **?** is used to display the onscreen help messages;
u: is the unit number, 0 through 7, of the Winchester disk unit that you wish to use;
p is the number of the Winchester partition you wish to assign to a drive name; and
d: is the drive name to which the Winchester partition will be assigned.

Preliminary Concepts

The ASSIGN utility enables you to access data from Winchester disk partitions by assigning the partitions to drives or by changing previous partition-to-drive assignments. ASSIGN can also be used to display all of a Winchester disk's current assignments.

Your Winchester disk can be divided into as many as four partitions to accommodate different operating systems and files. Under MS-DOS, all of these partitions can be made accessible at one time, provided they contain compatible versions of MS-DOS. Partitions are identified by partition numbers 1 through 4.

Although the boot partition is accessible as soon as you boot up from the Winchester disk, no other partition can be accessed until you run the ASSIGN utility to assign other partitions to drive names. If you boot from a floppy disk, no partitions are assigned for you.

For most MS-DOS operations, you can think of a Winchester partition as if it were a floppy disk. As such, you should realize that using the ASSIGN command is similar to the activity of inserting a floppy disk into a drive.

Just as you must insert a floppy disk into a drive before its data can be accessed, you must *assign* a partition to a drive name before it can be accessed (unless you booted up from that partition).

Command Line Entry

If you enter ASSIGN followed by a question mark (?) and press RETURN, your screen will display help messages. These messages include a short description of the ASSIGN command and its entry parameters.

For example, to display information on ASSIGN, you would enter

```
ASSIGN ?
```

and press **RETURN**. The ASSIGN help screen shown below would be displayed.

```
             ASSIGN Version 2.0


The ASSIGN utility is used to make a logical connection
between the partitions of a Winchester disk drive and the
drive names C:, D:, E:, and F:

Usage: ASSIGN [?][u:][u:p d:]

Where:

   ? - Print this list
   u: - Unit number of Winchester (0-7)
   p - Partition number to ASSIGN (1-4)
   d: - Drive letter (C: - F:)

If only [u:] is specified, a table of valid partitions
for drive [u:] is printed.
```

NOTE: The drive names in this example may differ from those that appear on your screen. The drive names that are displayed on the screen in help messages and prompts will reflect the specific hardware environment of your system. Thus, your system may be configured with two floppy disk drives and one Winchester, in which case the floppies would be drives A and B and the

Winchester would be C, D, E, and F; or you may have three floppy disk drives and one Winchester, in which case the floppies would be A, B, and C and the Winchester would be D, E, F, and G. However your system is configured, the onscreen messages will reflect that configuration.

To display the status of the Winchester partitions presently assigned, you would enter

```
ASSIGN u:
```

and press **RETURN**. The screen will display information about drive unit *u*: in a table such as the following:

```
            ASSIGN Version 2.0


   Partition        Start       End           Size in
     Type          Cylinder    Cylinder       Kilobytes
   ---------       --------    --------       ---------

1.   DOS              0          304            10369
2. unallocated        0           0                 0
3. unallocated        0           0                 0
4. unallocated        0           0                 0

  Drive C:  = 0:1
  Drive D:  = Unassigned
  Drive E:  = Unassigned
  Drive F:  = Unassigned

A>_
```

NOTE: This table is only an example; your system will probably be set up differently. However, the general layout and type of information displayed will be similar.

This table is similar to the one that is displayed when you use the PART program. (For more information on the PART command, refer to Chapter 17, "PART.")

ASSIGN

In order to assign partition 1 on drive unit 0 to drive C (as shown in the previous table), you would make the following entry

```
ASSIGN 0:1 C:
```

and press **RETURN**. Your screen will display the following message:

```
DOS partition 1 assigned to drive C.
```

You can now access DOS partition 1 as drive C.

NOTE: After a drive name has been assigned to a partition, that same drive name can be reassigned to a different partition at any time, by using the ASSIGN utility again.

Drives that have been assigned to partitions behave just as floppy disks. However, the following rules apply:

1. Partitions cannot be larger than 32 megabytes, or smaller than 32 kilobytes.
2. A single partition cannot be assigned to more than one drive name.
3. *u*: (the Winchester disk unit number) must be a number from 0 through 7.
4. *p* (the partition number) must be a number from 1 through 4.
5. *d*: must be a drive name that depends on the number of floppy disk drives in your system. The correct drive letter range is displayed when the command ASSIGN RETURN is entered.

   NOTE: The valid drive names for your particular system configuration will be listed in the help messages available by entering ASSIGN followed by a question mark.

Error Messages

```
Cannot communicate with Winchester controller
```

EXPLANATION: You will get this message if you try to run ASSIGN and you do not have a Winchester disk connected to your system.

```
Cannot read partition table
```

EXPLANATION: This message will be displayed if you try to assign a partition from a Winchester disk that does not exist.

```
Invalid drive/drive not available
```

EXPLANATION: You tried to assign a partition to a drive name other than those listed in the ASSIGN help screen. You must reenter assignments, using valid drive names.

```
Invalid partition selection
```

EXPLANATION: You tried to assign a partition number other than the valid partition numbers 1 through 4.

```
Invalid version of IO.SYS
```

EXPLANATION: You tried to assign a partition that contains a version of IO.SYS that is incompatible with the version you booted from. You must reboot from the partition in question or reboot from a floppy disk that contains a version of IO.SYS compatible with that on the partition.

```
Invalid Winchester drive number given
```

EXPLANATION: You must reenter the assignment with a valid Winchester drive number.

```
Not a DOS partition
```

EXPLANATION: If you try to assign a non-DOS (or unallocated) partition to a drive, you will get the above error message. You cannot assign non-DOS (or unallocated) partitions under MS-DOS.

ASSIGN

```
Partition is already in use
```

EXPLANATION: You tried to assign a partition number that is currently being used to a new drive name.

```
Partition is too large
```

EXPLANATION: As stated above, MS-DOS will support partitions of up to 32 megabytes. This error will occur if an attempt is made to assign a partition greater than the 32-megabyte limit. To remedy this error condition, you must run the PART program again and reallocate the partition space.

```
Partition is too small
```

EXPLANATION: The partition size allocated is less than the minimum DOS allocation (approximately 32K).

BACKUP (Transient)

Purpose

Use this command to create a single backup file from multiple source files. Each source file retains its unique identity within the backup file, and individual source files may be recovered from the backup file by using the RESTORE command.

Entry Forms

```
BACKUP
BACKUP ?
BACKUP [filespec[+filespec...]] [d:][filename][/x...]
```

where ***filespec*** is the file specification for a file or group of files to be backed up,
d: is the drive name of the destination drive on which the backup file will be written,
filename is the primary file name of the backup file, and
/*x* is one or more of the following switches:

/A[:*date*]	After date—Back up files dated after the specified date.
/B[:*date*]	Before date—Back up files dated before the specified date.
/D	Directory master—Locate all master backup files and give directory.
/E	Exception files—Exclude listed files from backup operation.
/F	Format silent—Format all destination disks without initial prompt.
/G	Global subdirectories—Back up files in all subdirectories as well as files in current working directory.
/L	List directory—List internal directory of the backup file.

/N	No formatting—Do not format destination disks.
/Q	Query each—Query "yes" or "no" on each source file before backup operation.
/R	Review selected files—Show list of files that will be backed up before beginning operation.
/T	Today's date—Back up files with today's date.
/V	Verify files—Verify backup files after copy.
/W	Written files only—Back up written files only.

Preliminary Concepts

Backing up files from your working disks, particularly Winchester disk partitions, can be tedious and time-consuming. The BACKUP command, however, provides a time-saving method for backing up routine work from any disk. The command may be used to back up from any readable device to any writable device except CON or PRN, as long as both devices can read and write both ASCII and binary characters. BACKUP is most useful when dealing with large numbers of files or when you wish to selectively back up files.

BACKUP collects groups of source files and copies them into a single destination file. This single file, called the *backup file*, can be larger than one disk. When a backup file is stored across more than one disk, each separate disk used to store the file is called a *volume* and is assigned a volume number. When more than one disk is required for the backup operation, the program will prompt you to change disks and will optionally format destination disks as they are required.

You provide the primary file name of the backup file created, and BACKUP automatically assigns a numeric file name extension. Extension assignments begin with .000 and are incremented sequentially if more than one disk volume is required to contain the backup file. For example, .000 will be assigned for the first disk, .001 for the second disk, .002 for the third, and so on. Volume numbers begin with 1 and are also sequentially assigned.

The BACKUP command may be invoked in two ways: interactive entry, wherein BACKUP prompts you for the required input; or command line entry, wherein you input the required parameters when you invoke the command. In addition, the BACKUP command provides a help screen display that briefly describes the BACKUP command and lists its syntax requirements and supported switches.

Normal events that occur during execution of the BACKUP command are prompts regarding the formatting of destination (backup) disks, prompts regarding the exchange of backup disks (where more than one disk is required), and a screen display of the name of each file that is backed up.

When BACKUP begins, and each time during program execution that a new backup disk is inserted in the destination drive, the system prompts

```
Format backup disk (Y/N) ?
```

NOTE: The appearance of the above prompt occurs when BACKUP is invoked *without* any of the optional switches affecting the formatting functions.

If you press **N** (for "no formatting") in response to the `Format backup disk (Y/N) ?` prompt, the backup operation will begin immediately.

If you press **Y**, the system prompts

```
Format single or double sided (S/D) ?
```

BACKUP

If the destination disk is single-sided, press **S**. The disk will be formatted accordingly. After disk formatting, the backup operation will begin.

If the destination disk is double-sided, press **D**. The disk will be formatted accordingly. After disk formatting, the backup operation will begin.

If you enter any value other than S or D, the preceding prompt will be displayed again.

During execution, BACKUP copies source files to the destination disk until execution is complete or until the disk is filled. Whenever a backup disk becomes full before execution is complete, BACKUP stops temporarily and prompts

```
Insert another disk in drive d for backup
and press RETURN when ready or any other key to abort.
```

At this point, you should remove the filled disk and replace it with a blank disk, then press RETURN to resume the backup operation. (When you press RETURN, BACKUP will normally prompt you whether or not to format the backup disk as described above.) However, if you do not want to continue with the backup operation, you can press any key other than RETURN and BACKUP will be aborted.

If more than one disk is required for the backup file, BACKUP must occasionally update and adjust the backup file's internal directory. When this is necessary, the system prompts

```
Insert backup master volume 1, filename.000, in drive d and
press RETURN when ready.
```

You should exchange disks as prompted and press **RETURN**.

During execution of BACKUP, the screen displays the file specification of each source file as that source file is copied to the backup file. This listing takes the following form:

```
A:FILENAM1.EXT
B:FILENAM2.EXT
B:FILENAM3.EXT
.
B:FILENAMn.EXT
```

BACKUP Help Screen

To obtain a screen display that summarizes the use, syntax requirements, and switches of the BACKUP command, enter

```
BACKUP ?
```

at the system prompt and press **RETURN**. The help screen is displayed as shown in Figure 11.1. The display is followed immediately by the system prompt, enabling you to refer to the information on the screen as you enter a BACKUP command line. More information on the various switches shown in Figure 11.1 is provided under Command Line Entry in this section.

```
              BACKUP version 2.xx


The BACKUP  utility is designed  to  take any number of source
files  and put them into a  single,  long file that may extend
across several volumes.  The source  file can originate on any
readable device. The backup destination is any device that can
be written to except CON and PRN.

Syntax: A:BACKUP [<filespec>[+<filespec>...] [<d:>]<filename>][</x>...]

     Switches: Default state: /F off, /Q off, and /V off.

/A AFTER date. /A:<mm-dd-yy>        /N NO formatting.
/B BEFORE date. /B:<mm-dd-yy>       /Q QUERY each.
/D DIRECTORY master.                /R REVIEW selected files.
/E EXCEPTION files.                 /T TODAY's date.
/F FORMAT silent.                   /V VERIFY files.
/G GLOBAL subdirectories.           /W WRITTEN files only.
/L LIST directory. <filespec>/l

A>
```

Figure 11.1. BACKUP Help Screen Display

```
                BACKUP version 2.xx

>
```

Figure 11.2. BACKUP Banner and Command Prompt

Interactive Entry Prompt and Response

If you enter **BACKUP** and press **RETURN** without entering any other information, the system will display the BACKUP banner followed by the BACKUP command prompt > as shown in Figure 11.2. When the command prompt is displayed, you may enter commands to BACKUP. When one command line has been executed, the command prompt will be displayed again and you may enter another BACKUP command line. BACKUP terminates *only* when you press RETURN or CTRL-BREAK at the command prompt without having entered a command line.

The command line entries that may be made at the BACKUP command prompt are the same as those described under Command Line Entry in this section, with the exception that you do not need to reinvoke BACKUP by beginning each line with the command name.

Command Line Entry

If you enter a complete, valid BACKUP command line at the system prompt and press RETURN, the single command line you entered will be executed and then the system prompt will be displayed again. If you wish to complete another backup operation, you must reinvoke BACKUP, unlike when using the interactive method described above.

You must always begin the command line with BACKUP and follow it with the required parameters and desired optional switches. The parameters of the BACKUP command line are described below. Entry requirements are noted, where appropriate, in the parameter descriptions.

Source File Specification

At least one source file specification, *filespec*, must be provided following **BACKUP** (or following the BACKUP command prompt if you use the interactive entry method). Wildcard characters (* and/or ?) may be used in the source *filespec* to designate more than one file.

If you wish, you may enter a series of source file specifications, as in *filespec*[+*filespec*...]. When a series of source file specifications is entered, each *filespec* in the series must be separated from the adjacent *filespec*(s) by a plus sign (+). There should be no spaces or delimiters other than the plus sign between the *filespecs*. Some or all of the source file specifications may include wildcard characters. Source files may be on the same or different disks. For each *filespec*, be careful to accurately enter the full file specification required by the system to locate the file. Note, however, that you may not include directory path names in the source file specification(s).

Destination File (Backup File)

Following the source file specification(s), you must specify the destination *filename* to which the backup file is to be written. If the destination file is on a disk other than the default disk, you must precede the *filename* with the appropriate drive name (*d*). The source file and the destination file parameters must be separated by a space.

You cannot use wildcard characters in the destination file name (only one destination or backup file can be created), and you should not specify a file name extension. BACKUP automatically assigns numeric file name extensions to the destination file name on the basis on the number of disks required to contain the file. On the first disk used, the backup file will have the extension .000; on the second disk (if used) the backup file will have the extension .001; and so on. The file name extensions are sequentially assigned for as many disks (volumes) as are required.

Switches

The use of any of the switches (/*x*) supported by the BACKUP command is optional. They are provided to allow you to tailor execution of the command to meet your needs. The switches are entered at the end of the command line and do not need to be separated (either from the preceding input or from each other) by a space. The switch character (/) is the only delimiter required.

If no switches are entered, BACKUP assumes the following operational defaults:

- query for formatting of destination disks—may be changed with /F or /N switch,
- do not query for each file name before backup—may be changed with /Q switch, and
- do not verify accurate copy after backup—may be changed with /V switch.

Supported switches are described individually below.

/A—After Date

Use the /A switch to back up all files that match the source file specification(s) and are dated (in the directory) *after* today's or the specified date. (The date recorded in the directory for any given file is called that file's *date stamp*.)

If you wish to back up files with a date stamp later than the current (today's) date displayed by the DATE command, enter the switch as

```
/A
```

at the end of the command line. All files dated after the date that was given to MS-DOS either when it was first booted up or when the DATE command was last used will be backed up. (BACKUP compares the directory date stamp for the specified file(s) to the date established by the DATE command.)

If you wish to specify a date other than the current date, enter the switch by using the format

`/A:mm-dd-yy`

where *mm* = a number from 1 to 12, signifying the month;
dd = a number from 1 to 31, signifying the day of the month; and
yy = a number from 00 to 99, signifying the year.

Note that the switch and the specified date must be separated by a colon (:) and that the only valid separator for the numbers in the specified date is a hyphen (-).

When the date is specified with the /A switch, BACKUP compares the directory date stamp for the specified source file(s) to the date you entered and backs up all files with a date later than the specified date.

/B—Before Date

Use the /B switch to back up all files that match the source file specification(s) and are dated (in the directory) *before* today's or the specified date. If you wish to back up files with a directory date stamp earlier than the current (today's) date displayed by the DATE command, enter the switch as

`/B`

at the end of the command line. All files dated before the date that was given to MS-DOS either when it was first booted up or when the DATE command was last used will be backed up. (BACKUP compares the directory date stamp for the specified file(s) to the date established by the DATE command.)

If you wish to specify a date other than the current date, enter the switch using the format

`/B:mm-dd-yy`

where ***mm*** = a number from 1 to 12, signifying the month;
dd = a number from 1 to 31, signifying the day of the month; and
yy = a number from 00 to 99, signifying the year.

Note that the switch and the specified date must be separated by a colon (:) and that the only valid separator for the numbers in the specified date is a hyphen (-).

When the date is specified with the /B switch, BACKUP compares the directory date stamp for the specified source file(s) to the date you entered and backs up all files with a date earlier than the specified date.

/D—Directory Master

Use this switch alone to check the default or a specified disk for *master backup files* and to display a master backup file directory. A master backup file is one with the extension ".000." In addition to copies of source files that were backed up, the master backup file includes the number of volumes (disks) on which the backup file resides, the number of files (sources) it contains, and the date it was made.

This switch must be entered by using the command line entry form

```
BACKUP [d:]/D
```

if the command is invoked at the system prompt; or the form

```
[d:]/D
```

if the command is invoked at the command prompt >. In either case, *d* is the name of the desired drive (if you want a master backup file directory for a disk other than that in the default drive). When you press **RETURN**, the screen displays a master

```
Name       Volumes   Files     Date      4-15-84

BACKFILE      3        10      12-19-83
BACK2         5       127       9-01-83
BACK3         1         1       8-22-83
```

Figure 11.3. Typical Master Backup File Directory

file directory similar to that shown in Figure 11.3. The date in the upper right corner of the display is the current date; all other information in the display pertains to the master backup files that are located on the default or specified disk.

/E—Exception Files

Use this switch to specify a file or files to be *excluded* from the backup operation. Exception files are listed immediately following the switch, in the form

/E:***filespec***[+***filespec***...]

Note that a colon (:) *must* be used between the switch (/E) and the exception file(s) listed. Wildcard characters may be used in some or all of the exception file specifications. When more than one exception file is listed, the file specifications must be separated by the plus sign (+). No spaces or delimiters other than the plus sign should be used between the *filespecs*.

When the /E switch is used, all files not listed after the switch that *do* match the source file specification(s) will be backed up. This switch is particularly useful when you want to back up many files, have used wildcard characters in the source file specification(s), and wish to exclude some files from the backup operation. An example is provided under Using the Exception Files and Query Each Switches in this section.

/F—Format Silent

As with any data or program disk, destination disks used by the BACKUP command must be properly formatted before they are used. Because this is so important, BACKUP normally is prepared to format each destination disk used before writing any backup files to it. The /F switch alters the default procedure for formatting backup disks.

Under default BACKUP operation, each time a new destination disk (including the first one used) is placed into the drive, BACKUP prompts you

```
Format backup disk (Y/N) ?
```

If you press **N**, the disk will not be formatted by BACKUP. If you press Y, BACKUP then prompts

```
Format single or double sided (S/D) ?
```

Press **S** if the disk is single-sided or **D** if the disk is double-sided. The disk is then formatted accordingly before the backup operation resumes.

When you use the /F switch, BACKUP assumes that *all* destination disks are to be formatted and will not display the `Format backup disk (Y/N) ?` prompt to give you a choice. It will, however, display the second prompt (`Format single or double sided (S/D) ?`) before formatting a new destination disk, and you must press either **S** or **D**.

/G—Global Subdirectories

Normally, BACKUP only backs up files in the current directory of the default or specified disk. Use the /G switch when you wish to back up files from all subdirectories as well as files in the current directory. The /G switch may be used when wildcard characters (such as *.*) are used in the source file specification(s) to indicate all files.

If the source file specification is *.* and the /G switch is *not* used, all files in the current directory will be backed up.

If the source file specification is *.* and the /G switch *is* used, then all files in the current directory of the specified disk and in all the subdirectories of that current directory will be backed up. If the current directory is the root directory, then all files on the disk will be backed up. If the current directory is a subdirectory, files in higher level directories will not be backed up.

/L—List Directory

This switch enables you to display a directory of the files that are contained within a specified backup file. The switch must be entered as part of a command line having the form

```
BACKUP [d:]filename/L
```

if entered at the system prompt, or

```
[d:]filename/L
```

if entered at the BACKUP command prompt >. In either case, *filename* is the primary file name of the backup file for which you wish to obtain an internal directory listing. The drive name (*d*) must be entered if the backup file is on a disk that is in a drive other than the default. Do not enter a file name extension for the backup file; BACKUP assumes the extension ".000."

When you enter a command line that includes the /L switch and press RETURN, the screen displays a listing of the individual files contained in the backup file, the device (drive name) from which each file was copied, the size of each file in bytes, the disk volume numbers on which the files are stored (one file may be stored across more than one disk volume), and the directory date stamp for each file. The display also includes entries for subdirectories (if any) and their parent directories. These are identified by <DIR> in the Start Volume column of the display. (Subdirectories will be included in a backup file *only* if the /G switch was used when the backup file was created.)

```
Drive    File Name    Date     Start   End     Size in
                               Volume  Volume   bytes

E:      TESTFIL1.DAT  10-10-82    1       1      3264
E:      TESTFIL2.DAT  10-10-82    1       2     19582
F:      TESTFIL1.DOC   8-01-82    2       3    887236
F:      TESTFIL2.DOC   9-17-82    3       3     22230
F:      DATA          10-10-82 <DIR>
F:      EXP1.DAT      10-10-82    3       3      1548
F:      EXP2.DAT      10-15-82    3       3      1675
F:      ..            10-10-82 <DIR>

6 file(s) on 3 volume(s)
```

Figure 11.4. Typical Backup File Internal Directory Listing

A typical internal directory listing is shown in Figure 11.4. Notice that a summary line at the bottom of the display gives the total number of files contained in the backup file, and the total number of volumes on which the backup file resides. Also, notice that the subdirectory (DATA in Figure 11.4) and the entry for its parent directory (..) are *not* counted as files in the backup file internal directory. The files that originated in the subdirectory are listed between the subdirectory name and the entry for the subdirectory's parent.

/N—No Formatting

Use this switch to suppress formatting of destination disks during the backup operation. When you use this switch, BACKUP will still prompt you to insert another backup disk when one volume is full but will *not* format the new disk. The `Format backup disk (Y/N) ?` and `Format single or double sided (S/D) ?` prompts are *not* displayed.

If you use this switch, you should be certain that the disks you plan to use for your backup file have been properly formatted before you begin BACKUP.

/Q—Query Each

When you use this switch, BACKUP will locate all files that match the source file specification(s) and prompt

```
Backup filespec (Y/N) ?
```

where *filespec* is a unique file specification for each file before beginning the backup operation.

The /Q switch is especially useful when you use wildcard characters in the source file specification and/or when you list a series of source files, because you are given the opportunity to review the list of files that will be backed up and delete any files that you do not want to back up. You cannot, however, add any files to the source files. If you have inadvertently omitted the file specification for a file you do want to back up, you will have to invoke a second backup operation when the first is completed.

If you *do* wish to back up a given file, press **Y** at the `Backup filespec (Y/N) ?` prompt for that file.

If you *do not* wish to back up a given file, press **N** at the prompt for that file; the file will be deleted from BACKUP's internal list of source files before the backup operation is begun.

/R—Review Selected Files

This switch is similar to the /Q switch, in that it enables you to review the files selected for backup before the backup operation is begun. However, the /R switch lists the entire group of files that match the source file specification(s) at one time, and you must accept or reject the entire group rather than individual files.

When you use the /R switch, BACKUP locates all files that match the source file specification(s), and then displays

```
            BACKUP version 2.0
```

```
Files to be backed up are:
```

This display is followed by a list of the file specifications and directories for files that will be backed up. Just below the list, the system prompts

```
Is this correct (Y/N)?
```

If the list is correct (that is, if it correctly shows all files that you wish to back up and does not include any files that you do not want to back up), press **Y**. The backup operation will begin.

If the list is not correct, press **N**. The backup operation will be aborted and the system or command prompt will be displayed.

/T—Today's Date

Use this switch to back up all files that match the source file specification(s) and are dated with the current date (as displayed by the DATE command). When you enter /T as part of the command line, BACKUP compares the original directory date stamp for the specified source file(s) to the date established by the DATE command. Then, only those files that have a matching date are backed up.

/V—Verify Files

Use this switch to cause BACKUP to verify the accuracy of each source file's backup copy immediately after the file is backed up. When you enter /V as part of the command line, BACKUP compares each source file with the destination file copy to make sure that there are no discrepancies. The file specification of each file is displayed during its verification in the form

```
filespec
Verifying filespec
```

If (and only if) an error is found, BACKUP prompts

```
Verify error, try BACKUP again (Y/N)?
```

If you wish to reattempt backup and verification for the file, press **Y**. The backup and verification operation will be repeated. If you do not wish to reattempt the backup operation, press N. The backup operation will be aborted for that file, and BACKUP will go on to the next source file.

If no verification errors are found, the screen simply displays the names of the files being backed up and verified.

/W—Written Files Only

Use this switch to back up written files only. *Written files* are those that have been changed since the last BACKUP operation. Whenever you revise an MS-DOS file, an archive bit in the directory entry for the file is set to 1 to show that the file has been changed. Whenever BACKUP is run for the file, the bit is reset to 0. When you enter /W as part of a command line, BACKUP checks each specified source file to see whether the archive bit is set to 0 or 1. If the bit is set to 1 (indicating that the file is written), then the file is backed up. If the bit is set to 0, the file is not backed up.

Specifying a Series of Source Files

You can use BACKUP to collect selected files from more than one disk and write them into a single backup file. For example, if you entered

```
BACKUP A:*.BAS+B:*.DOC+B:???X.COM+E:*.* BIGFILE
```

BACKUP would combine the following files in backup file named BIGFILE on the default disk:

- all files in the current directory of the disk in drive A that have the extension .BAS,

- all files in the current directory of the disk in drive B that have the extension .DOC,
- all files in the current directory of the disk in drive B that have a four-letter file name in which the fourth letter is X and that also have the extension .COM, and
- all files in the current directory of the disk in drive E.

Using the Exception Files and Query Each Switches

Suppose you have a working disk of letter, document, and data files—and that you wish to back up all files except the data files into the file named TXTBACK. Assume that letter files have the extension .LTR, document files have the extension .DOC, and data files have the extension .DAT. You could use the BACKUP command with wildcard characters in the source file specification and the /E switch to easily accomplish the backup operation.

Suppose the working files are on the disk in drive E and you wish to back them up onto a disk in drive A. You would enter the following at the system prompt:

```
BACKUP E:*.* A:TXTBACK/E:E:*.DAT
```

This would cause BACKUP to back up all letter and document files in the current directory of the disk in drive E, but data files would not be backed up. (If you wished to back up the letter and document files in all directory levels of the disk, you would have to make sure the current directory for the source disk was the root directory—and also use the /G switch as well as the /E switch.) The backup files would be written to TXTBACK on the disk in drive A.

Another way that you could selectively back up only the letter and document files from the working disk is by using the /Q (Query each) switch. Assuming the same circumstances as above, you could enter

```
BACKUP E:*.* A:TXTBACK/Q
```

In this case, BACKUP would be prepared to back up all files in the current directory of the disk in drive E and would prompt you for each file located, before the backup operation is begun:

```
BACKUP E:filename.ext (Y/N) ?
```

where *filename.ext* is a unique file. If you wished to back up the file named in the prompt (if it had a .LTR or .DOC extension), you would press **Y**. If you did not wish to back up the file named (if it had a .DAT extension), you would press **N**. When you had responded to the prompt for all files BACKUP found in the current directory, normal backup operation would begin.

Using the Interactive BACKUP Method

If you have a number of source disks containing a variety of files that you wish to selectively back up into more than one file, you may find it most convenient to use BACKUP in the interactive method. For example, suppose you have three working disks, each of which contains letter, document, and data files with the extensions .LTR, .DOC, and .DAT, respectively. If you wish to put all letter files into one backup file, all document files in another, and all data files in a third backup file, you could proceed as follows:

- At the system prompt, enter **BACKUP** and press **RETURN**. The BACKUP command prompt > will be displayed.

- To back up letter files from the disks in drives C, D, and E to the file LTRBACK on the disk in drive A, enter

  ```
  C:*.LTR+D:*.LTR+E:*.LTR A:LTRBACK
  ```

  and press **RETURN**. (In this and the following example steps, if drive A is the default, you do not need to enter the drive name for the destination file.) The backup operation proceeds normally. When all .LTR files have been backed up, the command prompt > is displayed again.

- To back up all document files from the disks in drives C, D, and E to the file DOCBACK in drive A, insert a new blank disk in drive A and then enter

  ```
  C:*.DOC+D:*.DOC+E:*.DOC A:DOCBACK
  ```

  and press **RETURN**. The backup operation proceeds normally. When all .DOC files have been backed up, the command prompt is displayed again.

- To back up all data files from the disks in drives C, D, and E to file DATBACK in drive A, insert a new blank disk in drive A and then enter

  ```
  C:*.DAT+D:*.DAT+E:*.DAT A:DATBACK
  ```

  and press **RETURN**. The backup operation proceeds normally. When all .DAT files have been backed up, the command prompt is displayed again.

- To exit BACKUP and return to the operating system, simply press **RETURN** when the command prompt is displayed.

Running BACKUP from a Batch File

If you create backups on a routine basis, you may find it advantageous to use BACKUP from a batch file. This automates most of the backup process and makes it even easier to make routine backups. An example is provided below; it may meet your needs, or may be tailored to fit your unique requirements. For more information on batch files, refer to Chapter 5, "Command Features."

NOTE: In the following batch file example, BACKUP is assumed to be on the default drive.

General Purpose Backup

If you do a lot of different types of work on a daily basis and you need to create backups of your work each day, you might try the following BACKUP batch file:

- Using a text editor or EDLIN, open a batch file named GENBACK.BAT and enter

  ```
  BACKUP %1*.* %2GENBACK/T/V
  ```

- Save the file GENBACK.BAT.

- When you are ready to back up your work, invoke the batch file by entering a command at the system prompt in the form

  ```
  [d:]GENBACK d: d:
  ```

  and pressing **RETURN**. In this entry, the first drive name (*d*) is the drive containing the disk with GENBACK.BAT on it, the second drive name is the drive on which the source files are located, and the third drive name is the destination drive for the backup file. The first drive name need not be entered if GENBACK.BAT is on the disk in the default drive.

This backup batch file creates a backup file named GENBACK.000 (and GENBACK.001, GENBACK.002, and so on, if the backup file requires more than one disk volume) that contains all the files in the current directory of the specified source disk that are stamped with the current date. In addition, the backup file is verified for accuracy.

Advanced Concepts

This section contains information that is not essential for you to use BACKUP or its counterpart, RESTORE. It does, however, describe some of the tasks that these commands perform that are beyond the concern or awareness of most users.

BACKUP makes backup files by placing all source files in a single file. This single file is a *master backup file* or, simply, a backup file. The backup file is a composite of one or more source files placed end to end. The backup file also includes an internal directory of all the source files it contains. The source files that are joined together into a backup file by BACKUP may be separated again into individual files by RESTORE.

Extension Assignment and Use of More Than One Backup Volume

Each backup file created by BACKUP has a primary file name that you assign. The primary file name must consist of from one to eight valid MS-DOS file name characters. BACKUP assigns its own three-digit extensions.

When files from a large capacity storage device (such as a Winchester disk partition) or from various storage devices are transferred to a smaller storage device (such as a 5.25-inch floppy disk drive) the source files can require more space than the smaller device has available. When the source files for a backup operation are collectively larger than the destination media, more than one disk is required for the backup file and a series of disks are used. The backup file in this case is actually a sequence of files with the same primary file name. Each of the files within the sequence has a different three-digit extension. An illustration of how BACKUP distributes source files across several disk volumes is provided in Figure 11.5.

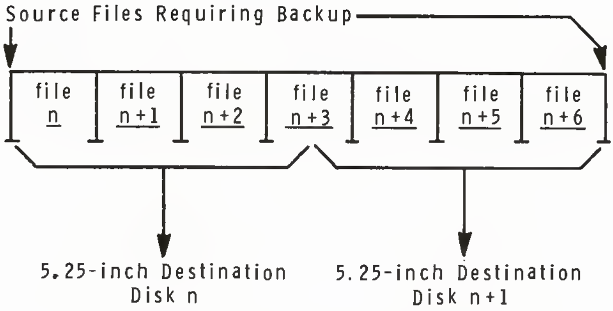

Source Files Requiring Backup
file n
file n+1
file n+2
file n+3
file n+4
file n+5
file n+6
5.25-inch Destination Disk n
5.25-inch Destination Disk n+1

Figure 11.5. Distribution of Source Files Across Several Backup Disks

BACKUP must keep track of all of the source files—even when the backup file is larger than the storage space available on one disk. To do this, BACKUP assigns a sequential extension to each file name, starting with the number 000 (which is the master volume and the backup file's beginning entry) and increases that extension by one number for each disk that the master file extends across. Each separate disk used to store a backup file is called a *volume* and is assigned a *volume number*.

The volume number is always the current file extension's value plus one. This way you can always tell which volume a backup disk is by using the DIR command. For example:

filename.000 is Volume 1
filename.001 is Volume 2
filename.002 is Volume 3
.
.
.
filename.nnn is Volume nnn + 1

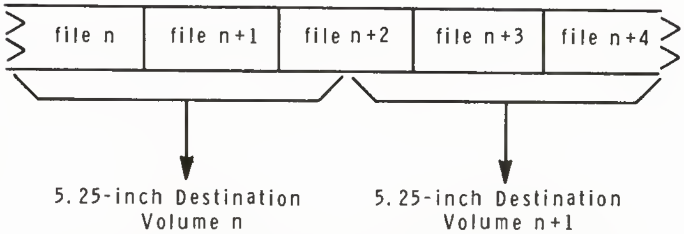
If a backup file is named BACKFILE and it extends across three disks, the DIR command shows the first disk's directory with the entry "BACKFILE.000" (which would be Volume 1), the second disk contains the entry "BACKFILE.001" (Volume 2), and the third contains "BACKFILE.002" (Volume 3).

BACKUP's Internal Directory

Suppose you have a 5.25-inch disk that runs out of available space midway through a backup operation and also midway through an individual source file. You need a way to keep track of both halves of the file. BACKUP provides a method for you to do this. BACKUP records this information in the backup file's internal directory by listing the volume on which the file starts on and the volume on which the file ends. This is illustrated in Figure 11.6.

The directory that is displayed after entry of a command with the /L switch (as described under Switches in this section) might be displayed as follows:

```
Drive   File Name      Date      Start    End     Size in
                                 Volume  Volume    bytes

E:     FILE1.DAT     10-10-82     1        1        3264
E:     FILE2.DAT     10-10-82     1        2       19582
F:     FILE3.DAT      8-01-82     2        3      887236
F:     FILE4.DAT      9-17-82     3        3       22230

4 file(s) on 3 volume(s)
```



	file n	file n+1	file n+2	file n+3	file n+4
Start Vol.	n	n	n	n+1	n+1
End Vol.	n	n	n+1	n+1	n+1

Figure 11.6. A Backup File's Internal Directory of Files and Volumes

The structure of a backup file, as mentioned earlier, is a directory followed by the source file or files that are being backed up. These source files are placed sequentially within the backup file. This arrangement requires only a little additional space for the backup file.

The directory is a composite of two types of entries. The first type of entry provides information about the backup file itself. The other type contains information about a file that has been backed up. The actual layout of the entries are as shown in Figures 11.7 and 11.8.

In a backup file directory entry such as that shown in Figure 11.7, the extension on the File Control Block (FCB) is used to record the number of volumes that the backup contains. The FCB also contains the date that the backup was created. The number of files is also entered here, as are the release and version numbers for compatibility with future versions. The ID byte for this entry is set to zero to distinguish this entry from normal file directory entries . and .. as shown in Figure 11.8.

The source file directory entry shown in Figure 11.8 is for a source file that is contained in the backup file. This entry contains information such as the file's name, starting and ending volumes, where on the disk volume the file starts, and the length of the file. The ID byte for this entry indicates the location of this source

ID	Extended FCB	4 res.	no. files	9 res.	Rel. no.	Ver. no.	6 res.	= 64 bytes
1	40	4	2	9	1	1	6	

Figure 11.7. Backup File Directory Structure

ID	Extended FCB	4 res.	start vol.	end vol	start posit.	no. bytes	9 res.	= 64 bytes
1	40	4	1	1	4	4	9	

Figure 11.8. Source File Directory Entry

file entry among the other source file entries in the directory. If the value of this byte is 2, then the entry describes a source file within the backup file. If the value of this byte is FF, then the entry is a "dummy" entry that merely signals the end of the backup file and does not describe a source file. In the source file directory entry, the attributes of the extended FCB are used to distinguish directories from files.

NOTE: Refer to Switches in this section for an illustration of a backup file internal directory listing that includes a subdirectory as well as source files.

Error Messages

```
Backup file name cannot not be ambiguous.
```

EXPLANATION: This message occurs if the wildcard characters ? or * were used in the backup file's name. A backup file must always be identified by a unique primary file name. Reenter the command line with a valid backup file name.

```
Cannot find master backup file filename.000.
```

EXPLANATION: This message occurs when you have requested an internal directory listing (by using the /L switch in a BACKUP command line) for a given master backup file (*filename*.000) that does not exist on the default or specified disk. Make certain that the correct disk is available and reenter the command line with the appropriate drive name.

```
Cannot open backup file filename.nnn, insert another
disk and press RETURN when ready, or press any other key to abort.
```

EXPLANATION: This message occurs when you are asked to insert volume nnn + 1 (which would contain *filename.nnn*) and the wrong disk is inserted. Insert the correct disk and then press **RETURN**.

```
Cannot open master backup file filename.000, insert another
disk and press RETURN when ready, or press any other key to abort.
```

EXPLANATION: This message occurs if the disk that has been inserted does not contain volume 1 of the backup file (*filename*.000). Insert the correct disk and then press **RETURN**.

```
Cannot open master backup file filename.000, not enough space
on disk.
```

EXPLANATION: This message will occur if there is less than 1 kilobyte of space available on the disk where the backup file is to be written. Try again, using a disk that is not as full or, preferably, a blank, formatted disk.

```
Conflicting switches /F and /N specified.
```

EXPLANATION: This message occurs if the switches for "Format silent" (/F) and "No formatting" (/N) were both requested. Reenter the appropriate command line with the correct switch.

```
Extension on backup file specified, extension 000 will be
assumed.
```

EXPLANATION: This message occurs whenever you try to assign an extension to a backup file name. (Backup file extensions are explained in the text entitled Advanced Concepts in this section.) If this occurs, BACKUP ignores the extension you requested and uses its standard, sequentially numbered extensions.

```
Format failure, insert another disk and press RETURN, or press
any other key to abort.
```

EXPLANATION: This message is displayed if an error occurs while trying to format a destination disk during BACKUP. This could be caused by the use of incorrect media or by damaged media. In either case, use another blank disk for the destination. Later you can run FORMAT on the disk that was to be the destination disk when the error message occurred to determine whether the disk's media is usable. If the disk cannot be formatted, you may have to discard it.

```
Insert another disk in drive d for backup, and press RETURN when
ready, or press any other key to abort.
```

EXPLANATION: This message occurs if the BACKUP destination disk in the specified drive (*d*) has no more space available on it. Insert another disk in the drive and press **RETURN** to continue the backup operation.

```
Invalid backup file.
```

EXPLANATION: This message occurs if the backup file specified in a BACKUP command line does not contain valid information. This may happen if the file specified was not a backup file but had an .000 extension, or if the data in a backup file has become degraded or inaccessible due to media damage. Possible causes for the latter include a bad disk sector or inadvertent exposure of the disk to an electromagnetic field.

```
Invalid date in switch.
```

EXPLANATION: This message occurs if the date given with the /A or /B switch was an invalid date or was not in the correct format. Reenter the command line, including a valid date in the proper syntax.

```
Invalid drive designation on BACKUP file.
```

EXPLANATION: This message occurs when a drive name is used that is not in the range of supported names (A: through H:) or that does not exist in your system.

```
Invalid exception file specifications.
```

EXPLANATION: This message occurs if you failed to specify at least one exception file following the /E switch or if exception files were specified with a syntax error. Reenter the BACKUP command line with the required /E switch parameter(s) in the correct syntax.

`Invalid file name.`

EXPLANATION: This message appears when a file name is specified that does not conform to the MS-DOS file naming conventions.

`Invalid selection file specifications.`

EXPLANATION: This message is generally caused by a typographical error in the command line. The message results when parameters in the command line appear garbled or incorrectly punctuated.

`Invalid switch /x specified.`

EXPLANATION: This message occurs if BACKUP is unable to recognize the switch that was specified in the command. Reenter the BACKUP command line with the correct switch.

`Invalid version of BACKUP for file filename.000.`

EXPLANATION: This message occurs if you use incompatible versions of the BACKUP program (that is, if the BACKUP program you are using is a different version than that used to create backup file *filename*.000). It may also occur if data in the backup file has become degraded or inaccessible due to media damage. Possible causes of this include a bad disk sector or inadvertent exposure of the disk to an electromagnetic field.

`No files selected.`

EXPLANATION: You get this message if your command line did not contain any valid file name, or if no files were selected because the files you specified for this operation were not stamped with the date(s) required by switches you entered. This message is also displayed if you enter N in response to every prompt when the /Q (query each) switch is used.

`Not enough parameters specified.`

EXPLANATION: This message results when the command to BACKUP is not complete enough for BACKUP to carry out the intended operations. Refer to Command Line Entry in this section, and enter a valid BACKUP command line.

```
Too many parameters specified.
```

This message will appear if you have entered more parameters than BACKUP can handle properly. Refer to Command Line Entry in this section, and reenter a valid BACKUP command line.

```
Verify error, try BACKUP again (Y/N) ?
```

EXPLANATION: This message occurs if a file does not verify correctly after a backup operation. You can choose to retry the file or skip the file and continue on to the next source file.

If you wish to retry the backup and verification operation for the file, press **Y**.

If you wish to skip the file in which the verification error occurred, press **N**. The backup operation will be begun for the next source file.

BREAK (Resident)

Purpose

The BREAK command allows you to tell MS-DOS to check for CTRL-BREAK (or CTRL-C) whenever an application program requests MS-DOS to perform any function, such as writing to disk.

NOTE: CTRL-BREAK and CTRL-C perform the same functions. In the remainder of this section, it is to be understood that any reference to CTRL-BREAK also applies to CTRL-C.

Entry Form

```
BREAK [ON]

BREAK [OFF]
```

where **ON** turns on the checking for CTRL-BREAK *anytime* a program makes a function request to MS-DOS; and

OFF instructs MS-DOS to check for CTRL-BREAK *only* during keyboard, screen, printer, or auxiliary device operations.

Preliminary Concepts

Use the BREAK command to specify when MS-DOS should check for CTRL-BREAK or CTRL-C being entered at the keyboard. The default setting of the BREAK command is set off. In this position it will only check for CTRL-BREAK being entered during keyboard, screen, printer, and auxiliary device operations.

If you are running an application program that uses CTRL-BREAK as one of its valid entry forms, you will want to turn off the BREAK command so that the MS-DOS CTRL-BREAK function is off. That way, when you press CTRL-BREAK within the application program, you affect your program and not the operating system. Specify BREAK OFF to turn off the MS-DOS CTRL-BREAK and BREAK ON when you have finished running your application program.

Command Line Entry

The BREAK command is entered by typing BREAK, followed either by ON or OFF, depending on whether you want to allow command interruption with CTRL-BREAK.

Entering BREAK with no parameters causes MS-DOS to display the current status (on or off) of the BREAK command.

For example, if you enter

```
BREAK
```

and press **RETURN**, the screen will display:

```
A>BREAK is off
```

(or BREAK is on, if that is the case.)

NOTE: The BREAK command is OFF at boot up.

Advanced Concepts

You can also set BREAK ON by default (turn on the extended checking) by inserting

```
BREAK=ON
```

in your configuration file. Refer to Chapter 9, "System Component Features."

CHDIR or CD (Resident)

Purpose

The CHDIR (change directory) command is used to change the current working directory to a different path or to display the current working directory.

Entry Forms

```
CHDIR [d:][pathname]
CD [d:][pathname]
```

where ***d*** is the letter of the designated disk drive; and
pathname is a sequence of characters of the form:

[\] [***directory***] [***directory...***]

Optionally you may use the MS-DOS shorthand notation shown below in lieu of *directory*:

• MS-DOS uses this shorthand notation to indicate the name of the current working directory in all directory listings. MS-DOS automatically creates this entry when a directory is created.

• • MS-DOS uses this shorthand notation to indicate the name of the current directory's parent directory. MS-DOS automatically creates this entry when a directory is created.

NOTE: These two shorthand symbols do not exist in the root.

Preliminary Concepts

The CHDIR command allows you to *travel* to different branches of the directory tree. It can also be used to display the path name of the current working directory. You may use the synonym CD, instead of the letters CHDIR, as an abbreviation. For more information on how the MS-DOS directory structure of MS-DOS works refer to Chapter 7, "Directory Features."

Command Line Entry

The CHDIR command is primarily used in three ways:

- to change the current working directory to another directory.

- to quickly put you into the parent directory of your current working directory, and
- to display the path name of the current working directory.

The root directory may be accessed from any file, or directory, by entering

```
CHDIR \
```

at the system prompt and pressing **RETURN**.

This will convert the root into the current working directory.

Specifying the New Directory by Name

If you wish to change the current working directory to another directory, follow the example below. Remember that the backslash character (\) is used to denote the *root*, or to separate directory names.

If the current working directory of the default disk is \BIN\USER\JOE and you want to change to another directory (such as \BIN\USER\JOE\FORMS), enter

```
CHDIR \BIN\USER\JOE\FORMS
```

at the system prompt and press **RETURN**. This command will place you in the FORMS directory (that is FORMS becomes the current working directory of the default disk).

This example utilizes an *absolute path name*, that is one that begins with the root (\). There is, however, another type of path name called a *relative path name*. This type of path name does not begin at the root, but begins at the current working directory. The next example will move from the current working directory \BIN\USER\JOE, as in the previous example, but will use a relative path name to do so. Assuming that you are still in the \BIN\USER\JOE directory, enter

```
CHDIR FORMS
```

and press **RETURN**. This command will also place you in the FORMS directory.

If you use the alternate name CD, in place of CHDIR in the previous examples, the outcome would be identical.

Specifying the New Directory by Shorthand Notation

There is another method you can use to execute the CHDIR command. However, this particular variation will only move you from the current working directory to its parent directory.

For example, if you wish to change from the current working directory to the parent of the current working directory, use the shorthand notation by entering

```
CHDIR ..
```

at the system prompt and pressing **RETURN**. This command will always put you in the parent directory of your current working directory, unless your current directory is the root. If the current directory is the root, then the error message

```
Invalid directory
```

will be displayed. Remember, the • and • • notations for the current and parent directories, respectively, may not be used in the root directory.

Displaying the Status of the Current Working Directory

If you wish to see the path name of the current working directory displayed, enter CHDIR without any parameters. Doing this will cause MS-DOS to display the path name of your current working directory. If your current working directory is \BIN\USER\JOE on drive B, enter

```
CHDIR
```

and press **RETURN**. Your screen will display:

```
B:\BIN\USER\JOE
```

This command is useful if you want to know the path name of your current working directory.

Error Message

```
Invalid directory
```

EXPLANATION: This message will be displayed if you enter an invalid directory name, such as two backslashes in a row (\\) or if you specify a nonexistent directory within the command line. Reenter the command.

CHKDSK (Transient)

Purpose

The CHKDSK command (Check Disk) is used to examine the directory of the disk in the default or designated drive, provide information about the disk's contents, and fix disk problems under certain circumstances.

Entry Form

```
CHKDSK [D:][filename][/x]
```

where ***d*** is the letter of the designated drive;
filename is the name and extension of the designated file;
/x is one of the following optional switches:
  /F is the fix switch; and
  /V is the switch that causes CHKDSK to display messages and a directory during execution.

Preliminary Concepts

CHKDSK scans the directory of the disk in the default or designated drive and checks it for consistency and errors. CHKDSK accomplishes this by analyzing the directory and the File-Allocation-Table (FAT) of the specified disk. It then produces a status report of any inconsistencies, such as files which have a non-zero size in their directory entry, but contain no data. CHKDSK should be run on each disk occasionally to verify the integrity of the directory structure. If any errors are detected, the appropriate error message is displayed, along with prompts so that corrective action can be taken.

CHKSDK will not correct the errors found in your directory unless you specify the /F (fix) switch. Entering the /V switch causes CHKDSK to display messages and the directory while it is running.

CAUTION: Be sure to use a version of CHKDSK that matches the version of MS-DOS you are checking. That is, you must not run versions of CHKDSK that are verson 2 or above on versions of MS-DOS (or Z-DOS) that are below version 2, or vice versa. If you do not follow this rule, your data could be scrambled or even erased. Refer to the text entitled Error Messages in this section.

Command Line Entry

The CHKDSK command may be entered without parameters, to check the disk in the default drive, or entered followed by these parameters:

1. The letter of the drive which contains the disk you wish checked (unless you wish to check the default drive).

2. The file name whose status you wish to check. This may cause CHKDSK to return the additional message:

   ```
   All specified file(s) are contiguous
   ```

   This means that the specified file's clusters (the physical groups that make up the file) are in consecutive order on the disk with no other files intervening in that file's allocation area.

3. The /F (fix) switch, which will automatically correct the most common errors discovered by CHKDSK.

4. The /V (verbose) switch, which causes messages and the directory to display while CHKDSK is running.

When CHKDSK is executed, any errors that are found are displayed via error messages, followed by a status report. The status report indicates:

- Total disk space (size of the disk, in bytes).
- Number and size of hidden files.
- Number and size of directories (**NOTE**: no entry for number of directories if the root directory is the only directory).
- Number and size of user files.
- Number of bytes in bad sectors (**NOTE**: this will only appear if there is data in bad sectors).
- Amount of free space remaining on the disk.
- Size of total memory.
- Amount of memory available for program execution.

Examples

The following example shows what would be displayed if you ran CHKDSK on a fictional disk in drive B. You would enter the following command at the system prompt

```
CHKDSK B:
```

and press **RETURN**. The screen would display the following information.

```
Volume (LABEL) created JAN 7, 1984 10:30a

160256   bytes total disk space
  8192   bytes in 2 hidden files
   512   bytes in 2 directories
 30720   bytes in 8 user files
   104   bytes in bad sectors
121240   bytes available on disk

 65536   bytes total memory
 53152   bytes free
```

NOTE: If no label exists, the `No Label` message will appear at the top of this display.

If there are any errors, the appropriate error message will appear during the CHKDSK process.

Another way to enter CHKDSK is to specify a file name in the command line along with the other parameters.

For example, you could enter all of the information from the previous example, but add a file specification such as

```
CHKDSK B:DIARY.ANN
```

and press **RETURN**. The screen would display the same information as before, with the addition of the message

```
All specified file(s) are contiguous
```

or, if the specified file is not found, the screen would display

```
B:\DIARY.ANN
  File not found
```

Advanced Concepts

CHKDSK is useful to verify that the current disk's contents and the disk directory do indeed coincide. CHKDSK can resolve many of the problems that it finds. These problems might cause errors if left unchecked.

Pressing **CTRL-PRTSC** *and* **CHKDSK** and then pressing **RETURN** sends a copy of the status report to your printer. This may then be cut to size and attached to the disk sleeve for reference. This may also be done with the DIR (Directory) command. However, if there are a lot of files on the disk, the listing produced could prove too lengthy to attach without folding.

In addition, you can redirect the output from CHKDSK to a file. To perform this function, enter

```
CHKDSK A:>filename
```

and press **RETURN**. Any errors found by CHKDSK will be sent to the file you specified in the command line.

NOTE: Do not use the /F switch if you redirect CHKDSK output, because it will automatically "fix" some things, and so will not give a complete list of the errors.

You may see the following message displayed when you run CHKDSK:

```
filename contains
non-contiguous blocks
```

This is not an error message, but an informational message describing the condition of the *filename* listed. It means that the file in question is not written sequentially (contiguously) on disk.

This message is to alert you to this information. You may wish to copy a noncontiguous file to another disk, because copying records files contiguously. Note that files recorded noncontiguously can take much longer to read.

Error Messages

If an error is detected and the /F switch is not specified, CHKDSK will display the error message. If the /F switch *is* specified, only the errors not fixed will be displayed. If the /V switch is specified in addition to the /F switch, all errors encountered will be displayed, including those fixed.

The following errors will be corrected automatically if you specify the /F (fix) switch:

```
Allocation error, size adjusted
```

```
Entry has a bad link (or size or attribute)
```

NOTE: Only one will display.

```
First cluster number is invalid
entry truncated
```

```
Has invalid cluster, file truncated
```

You must correct the following errors displayed by CHKDSK, even if you specified the /F switch:

```
Cannot CHDIR to root
Processing cannot continue
```

EXPLANATION: The disk you are checking is bad. Restart the system and run RECOVER on the disk.

```
Cannot CHDIR to filename
Tree past this point not processed
```

EXPLANATION: CHKDSK is unable to continue processing any of the directory subtree past *filename* because the disk is bad. You must run RECOVER.

```
Directory is totally empty, no . or ..
```

EXPLANATION: CHKDSK found a bad directory. It will attempt to correct the problem if the /F switch was specified. One or possibly both of the following mesages might appear:

```
Cannot recover . entry, processing continued
```

EXPLANATION: The directory in question is bad and cannot be recovered.

```
Disk error reading FAT
```

EXPLANATION: The disk you are checking is bad. Restart the system and run RECOVER on the disk.

```
Disk error writing FAT
```

EXPLANATION: The disk you are checking is bad. Restart the system and run RECOVER on the disk.

```
Disk error writing FAT x
```

EXPLANATION: A disk error occurred while CHKDSK was trying to update the File-Allocation-Table (FAT). The variable *x* will be a 1 or a 2, depending on which copy of the FAT could not be written. If this message appears for both FAT's, the disk is unusable.

```
Errors found, F parameter not specified
Corrections will not be written to disk
```

EXPLANATION: You did not specify the /F switch. However, CHKDSK will complete an analysis as though corrections were going to be made. This allows you to see the results of the analysis, but no corrections will be written to disk. You must specify the /F switch if you want these errors corrected by CHKDSK.

```
File allocation table bad
Drive A:
```

EXPLANATION: The disk you are checking is bad. Restart the system and run RECOVER on the disk.

```
filename is cross linked on cluster
```

EXPLANATION: This message should appear twice, once for each of the cross linked files. To take corrective action, follow the steps below:

1. Make copies of both files, using the COPY command.
2. Erase the original files, using the DEL command.
3. Examine the files to verify their integrity.

```
Incorrect DOS version
```

EXPLANATION: You cannot run this version of CHKDSK on versions of MS-DOS (or Z-DOS) that are not version 2 or above.

```
Insufficient memory
Processing cannot continue
```

EXPLANATION: There is not enough memory in your microcomputer to run CHKDSK on this disk. You must obtain more memory to run CHKDSK.

```
Insufficient room in root directory
Erase files in root and repeat CHKDSK
```

EXPLANATION: CHKDSK cannot write the recovery files, FILE*xxxx* to the root until you delete unneeded files from the root directory.

```
Invalid drive specification
```

EXPLANATION: You specified an invalid drive specification to CHKDSK. Reenter the command line.

```
Invalid current directory
Processing cannot continue
```

EXPLANATION: Restart the system and run CHKDSK again.

CHKDSK

```
Invalid parameter
```

EXPLANATION: You specified an invalid parameter to CHKDSK. Reenter the command line:

```
Invalid subdirectory entry
```

EXPLANATION: CHKDSK discovered invalid information in the named subdirectory. CHKDSK will try to correct the error. For more information on the nature of the error, run CHKDSK again with the /V switch.

```
Probable non-DOS disk
Continue (Y/N)?
```

EXPLANATION: The disk you are using is probably a non-DOS disk. If you did not use the /F switch, and you reply Y to this prompt, CHKDSK will show you what corrective actions are possible without actually completing them. If you did use the /F switch, and reply Y, CHKDSK will take the indicated corrective action. You must indicate whether or not you want CHKDSK to continue processing.

```
Unrecoverable error in directory
Convert directory to file (Y/N)?
```

EXPLANATION: If you respond Y to this prompt, CHKDSK will convert the bad directory into a file. You can then examine it with DEBUG, or delete it. If you respond N to this prompt, the entry remains the same.

```
x lost clusters found in y chains
Convert lost chains to files (Y/N)?
```

EXPLANATION: If you respond Y to this prompt, CHKDSK will create a directory entry and a file for you to resolve this problem. Files created by CHKDSK are named FILE*nnnn*.CHK, where *nnnn* is a sequential number starting with 0000. These special files are created and stored in the root directory of the designated drive. You should examine them to see if they contain needed information; if not they can be erased. CHKDSK will then display:

```
   x bytes disk space freed
```

NOTE: You must have specified the /F switch for this remedial action to work.

If you respond N to this prompt and have not specified the /F switch, CHKDSK frees the clusters and displays:

```
x bytes disk space would be freed
```

CIPHER (Transient)

Purpose

Used to encrypt and decrypt files for security purposes, with execution of the encryption or decryption operation dependent upon a user-defined keyword.

Entry Forms

```
CIPHER keyword >filespec
CIPHER keyword <filespec
CIPHER keyword <filespc1 >filespc2
```

where ***keyword*** is the user-defined keyword required to encrypt or decrypt the specified file;
> directs the output of the CIPHER command to the file specified (that is, the *filespec* immediately following >);
< directs CIPHER to read the file specified for input (that is, the *filespec* immediately following <);
filespec is the file specification of the file to be encrypted or decrypted;
filespc1 is the file specification of an encrypted file that you wish to decrypt; and
filespc2 is the file specification of the file to which you want the decrypted file written.

Preliminary Concepts

The CIPHER command is a useful and flexible tool enabling you to create encrypted files, read encrypted files, create encrypted copies of regular files, and decrypt files. Encryption and decryption require a user-defined *keyword* for completion of the CIPHER operation. Once a file has been encrypted, the same keyword must be used to decrypt the file as was used to encrypt it.

You can encrypt any file that you wish to be unavailable to other users. If a program file is encrypted, it must be decrypted before it can be executed. If a text or data file is encrypted, it must be decrypted before it can be read or accessed by any program. Without the proper keyword and its correct use in a CIPHER command line, encrypted files cannot be displayed on the screen, nor can they be printed in an understandable form.

Command Line Entry

The parameters of the CIPHER command line are described below. The use of the various parameters varies depending upon the operation being invoked. Examples of CIPHER command lines are provided later in this section.

User-Defined Keyword

A user-defined *keyword* must be entered immediately following the command name when you are encrypting or decrypting a file. CIPHER uses the keyword to allow or deny access to a file; once a file has been encrypted using a given keyword, only that keyword may be used to display file contents or to decrypt the file.

A keyword may be any alphanumeric word or string of any length desired, as long as the entire command line does not exceed the input buffer maximum of 127 characters. The keyword itself must not contain any spaces (or equivalent MS-DOS delimiter);

however, the keyword must be separated from the command name and other command line parameters by spaces. (If you enter a space within a keyword, only the characters preceding the space will be seen by CIPHER as being a keyword. Any characters after the space will be ignored.)

Any letters you enter as part of a keyword are *not* automatically converted to uppercase. Thus, in order for CIPHER to recognize a keyword in subsequent use, it must always be entered *exactly* as it was entered when first used to encrypt a file.

Source and Destination File Specifications

Conventional MS-DOS file specifications, or *filespecs*, are used to identify the file being created, encrypted, or decrypted, and to identify the source and destination files when you are copying files using CIPHER. If the file specified is in the current directory of the disk in the default drive, the *filespec* may consist simply of the primary file name and extension. If the file specified is in another directory and/or on another disk, you must also include the drive name and/or appropriate path name in the file specification.

Input/Output Redirection

Input and output redirection are used extensively in the CIPHER command, as shown under Entry Forms above. You may use > to direct keyboard/screen display output to a file when creating an encrypted file, use < when using CIPHER to display an encrypted file, or use both input (<) and output (>) redirection when copying files using CIPHER. These uses of input/output redirection are explained in the examples provided in this section. For more information on the redirection of input and output, refer to Chapter 8, "Input/Output Features."

Encrypting an Existing File

Suppose you have a program, data, or text file created with a text editor or EDLIN, and you wish to encrypt the file to prevent unauthorized access or use of the file. CIPHER easily enables you to protect the security of your file. For example, suppose you have a data file, WTRGATE.BRK, containing compromising information, and wish to make certain that the file is not read by anyone else. Using CIPHER, you may copy and encrypt the file, then delete the original file, thus protecting the secrecy of the file's contents. Assume that CIPHER and WTRGATE.BRK are on the disk in the default drive. To copy and encrypt the file to drive C, enter

```
CIPHER RMN <WTRGATE.BRK >C:FLUFF.DAT
```

and press **RETURN**. In this command line, **RMN** is the keyword by which CIPHER will allow or deny access to the encrypted file, and **C:FLUFF.DAT** is the destination file specification to which the encrypted file will be written.

When you press **RETURN**, the encryption and copying process begins. The source file (WTRGATE.BRK) contents are encrypted and written to file FLUFF.DAT in the current directory of the disk in drive C. When the operation is complete, the system prompt will be displayed again. If you display a directory of the source and destination disks, you will see that the source file still exists. To keep only the encrypted file, use the DEL or ERASE command to delete the WTRGATE.BRK file from the default disk.

Creating an Encrypted File

You may use CIPHER to create and encrypt a file in one operation as well as to encrypt an existing file. To do this, you use the entry form

```
CIPHER keyword >filespec
```

where ***keyword*** is a keyword you define;

> directs the output of the CIPHER command to the file specified; and

filespec is the file specification for the file you want to create.

Using CIPHER in this way enables you to write information or data input via the keyboard (and displayed on the screen) to an encrypted disk file.

For example, suppose you wished to create an encrypted text file named SECRET.TXT in the current directory of the default disk, using the keyword HUSH. Rather than create the file by using a text editor program or EDLIN and then encrypting it, you could create and encrypt it in one operation, as follows:

- At the system prompt, enter **CIPHER HUSH >SECRET.TXT** and press **RETURN**. The command line is displayed on the screen, and, after a short interval of disk activity, the cursor is displayed on the line below the command.

  NOTE: This example and others in this section assume that CIPHER is on the disk in the default drive. If it is not, you must precede the command name with the appropriate drive name.

- Type in the text you wish the file to contain. End each line with a **RETURN**. It is not necessary to use CTRL-ENTER to create a line feed. While you type the text, all standard MS-DOS editing and control keys may be used. (Refer to Chapter 5, "Command Features," for information on the editing and control keys.)

- After the end of the text (on the blank line following the last line of the file), press **F6** and then press **RETURN**. (This enters the end-of-file character, 1AH, in the file and ends the input operation.) The text you entered will be encrypted and written to the file SECRET.TXT, then the system prompt will be displayed again. At a later time, you may decrypt the file and display it on the screen. When an encrypted file is displayed in decrypted form, you may print it if you wish by using the CTRL-PRTSC function described in Chapter 8, "Input/Output Features."

Displaying an Encrypted File

To display an encrypted file on the screen, use the entry form

```
CIPHER keyword <filespec
```

where ***keyword*** is the keyword originally used to encrypt the file;
filespec is the file specification of the file you wish to display; and
< causes CIPHER to read from the specified file.

NOTE: If you used a > bracket instead, CIPHER would encrypt keyboard input and write it to the file specified, as described under Creating an Encrypted File.

For example, suppose you have the file SECRET.TXT in the current directory of the default disk and that the file was created as described under Creating an Encrypted File. To display the contents of the file on the screen without altering the file's contents or permanently decrypting the file, enter

```
CIPHER HUSH <SECRET.TXT
```

and press **RETURN**. Note that to display an encrypted file, you *must* enter the same keyword as was used to encrypt the file. Otherwise, the screen will display meaningless characters.

Copying an Encrypted File to Another File

Once a file has been encrypted, you may use the COPY command to copy the file, in encrypted form, to another file. For example, suppose you have the encrypted file CIASPY.DAT on the disk in drive B and that you wish to move the file's encrypted contents to another file named BROTHER.DAT on the same disk.

To use the COPY command for this task, enter

```
COPY B:CIASPY.DAT B:BROTHER.DAT
```

at the system prompt and press **RETURN**. When the copy operation is complete, the system prompt will be displayed again. The source file is not affected; its contents are simply duplicated in the destination file. (Refer to COPY in this chapter for more information on the COPY command.) If you wished, you could delete the source file by using the DEL or ERASE command.

The destination file created in this way may be decrypted using the same keyword as was used to encrypt the source file.

Decrypting a File

To decrypt a file and write the decrypted file to a specified file, use the entry form

```
CIPHER keyword <filespc1 >filespc2
```

where ***keyword*** is the user-defined keyword used to encrypt the file you are going to decrypt;
< directs CIPHER to read the specified encrypted file;
filespc1 is the file specification of the encrypted file;
> directs the output of CIPHER to the specified destination file; and
filespc2 is the destination file to which you want the decrypted file (that is, the output of this CIPHER operation) written.

For example, suppose you have the encrypted file NOSEEUM.DAT on the disk in drive B and you wish to decrypt the file in order to use it in a program you are running. Assuming that the file was encrypted using the keyword OMAR, decrypt the file by entering

```
CIPHER OMAR <B:NOSEEUM.DAT >B:SEEUM.DAT
```

and pressing **RETURN**. This causes CIPHER to decrypt the contents of the file B:NOSEEUM.DAT and write them to the file B:SEEUM.DAT. The original encrypted file will not be affected. You could then use the file SEEUM.DAT in the program you were running, and either encrypt it or delete it when finished.

Error Message

```
Must specify a code word
```

EXPLANATION: This message is displayed if you did not enter the required keyword when invoking the CIPHER command to encrypt, display, or decrypt a file. Reenter the command line, including in it the required keyword.

CLS (Resident)

Purpose

The CLS (CLear Screen) command clears your microcomputer's screen.

Entry Form

```
CLS
```

Preliminary Concepts

The CLS command causes MS-DOS to clear the screen on your microcomputer and print the MS-DOS prompt at the top left corner of the screen.

Command Line Entry

The CLS command is entered by typing:

```
CLS
```

and pressing **RETURN**.

COMMAND (Transient)

Purpose

Enables you to use the features of COMMAND.COM as a transient command. This function of COMMAND is most useful when invoking a command interpreter (such as COMMAND.COM) from within another program by using EXEC system calls.

Entry Form

```
COMMAND [d:][pathname] [cttydev] [/x]
```

where *d*: is the specified drive name (current default drive, if none specified);
pathname is the specified directory on drive *d*. The root directory (\) is the default, if none specified;
cttydev is the name of the CTTY device. The current CTTY device is the default (this will be CON in most cases). If the command is permanent (/P is specified), the *cttydev* will be \DEV\CON. The \DEV\ may be omitted if the statement, AVAILDEV = TRUE, is included in CONFIG.SYS; and
/x is one of the following optional switches;
  /D is used to instruct COMMAND.COM to *not* prompt for the time and date;

/P is used to make COMMAND.COM permanent in memory; otherwise, the default makes COMMAND.COM transient; and

/C is entered in the form:

`[/C string]`

where the /C switch must be the last entry in the command line and *string* is any character string (up to 128 characters, total line length).

Preliminary Concepts

The use of COMMAND.COM as an executable command presupposes a certain level of knowledge about MS-DOS. COMMAND.COM is the command processor supplied with MS-DOS.

The COMMAND.COM file is not given hidden file status. Therefore, you can *view* it with the DIR command, *copy* it with the COPY command, or *delete* it with the DEL command. Furthermore, COMMAND.COM can be recorded on any disk area that is available beyond MSDOS.SYS.

COMMAND.COM is sometimes referred to as the *command interpreter*. In general, COMMAND.COM acts as the interface between your input and the other components of the operating system. COMMAND.COM provides this interface by interpreting the commands that you enter.

You can also use a command interpreter other than COMMAND.COM as a system component. To use a different command interpreter, specify it in a CONFIG.SYS file in the root directory of the booted media, reset, and reboot.

COMMAND.COM consists of three basic parts:

1. A resident portion that is always in memory unless another file is overlaid on its space, or you choose to load another command processor of your own construction. This part of COMMAND.COM is responsible for all error handling by the system and the displaying of errors.

The resident portion also enables the system to load programs and interrupt program execution when you enter CTRL-BREAK or CTRL-C. This portion remains resident in memory from the time you boot up the system until the time you reset the system.

2. An initialization portion is loaded into memory immediately after the resident portion.

   This section contains the AUTOEXEC.BAT file processor setup routine. The initialization portion determines the segment address at which programs can be loaded. It is overwritten by the first program that COMMAND.COM loads.

3. The third portion of COMMAND.COM is the *shell*, or the *user interface*. This is the part of COMMAND.COM that you see when you use the system. Usually, this will be the system prompt (`A>`), that reads and processes the commands you enter at the keyboard. This portion, sometimes known as the *transient portion*, contains all of the MS-DOS resident commands, that is, DIR, PATH, and TYPE. For transient commands, this portion of COMMAND.COM builds a command line and issues an EXEC function request to load the program and transfer control to it.

   This transient portion contains the resident command processor and the batch processor. It also reads commands entered through the keyboard (or batch file) and causes those commands to be executed.

   When a transient command (other than COMMAND.COM) is loaded for execution, this command may overwrite the transient portion of COMMAND.COM if it needs the memory space. Therefore, the transient portion of COMMAND.COM may have to be reloaded from a disk or partition containing the COMMAND.COM file after the transient command is executed.

For more information about the system component aspects of COMMAND.COM, refer to Chapter 9, "System Component Features."

Command Line Entry

COMMAND.COM can be entered in different ways, which reflect the ways you wish to use it. If you enter the following command at the system prompt

```
COMMAND
```

and press **RETURN**, COMMAND.COM will reload itself into memory, and display the default system prompt.

If you wish to load COMMAND.COM into memory permanently (that is, it will not be moved from memory except by rebooting), you can specify the /P switch at the end of the command line.

COMMAND.COM will prompt you for the date and time by default. If you wish to set this function off, then specify the following command line:

```
COMMAND /D
```

and press **RETURN**. This will instruct COMMAND.COM not to prompt you for the date and time.

You may also instruct COMMAND.COM to execute a string as if you had typed it at the system prompt. This is accomplished by entering

```
COMMAND /C string
```

and pressing **RETURN**. The variable, *string*, may be any character string up to a total line length of 128 characters. COMMAND.COM will execute this string (if it is executable) and then exit. If the /P switch is present, it is ignored because you cannot have a permanent COMMAND that executes a single command.

NOTE: All text on the command line following the string specified in the /C switch is ignored. It is not processed for more arguments; therefore, the /C switch must be the last item on the command line.

Advanced Concepts

One way in which COMMAND.COM becomes useful as an executable command is when it is accessed by the APPLY command. In order to use the APPLY command you must have a valid version of COMMAND.COM on the disk in the default drive.

APPLY is used to execute a resident command while you are running another program. Ordinarily you could not execute a resident command in this situation, because resident commands cannot be executed without first loading COMMAND.COM. If COMMAND.COM is not loaded, only transient commands (.COM, .EXE, and .BAT) can be executed.

However, when you use APPLY to EXEC COMMAND.COM, COMMAND.COM executes the intended resident command sequence.

NOTE: For more information on the operation of the APPLY command, refer to the section entitled, APPLY, in this chapter.

Another use for COMMAND.COM as an executable command is when you alter the distribution version of COMMAND.COM with a CONFIG.SYS file. If you are creating a CONFIG.SYS file, you have two options that affect the use of COMMAND.COM. These are the two CONFIG.SYS statements:

1. SHELL = filename,
2. AVAILDEV = TRUE or AVAILDEV = FALSE.

The first of these lines enables you to load an alternate command interpreter. For example, if you entered the line

```
SHELL = MYCOM.COM
```

in your CONFIG.SYS file, where **MYCOM.COM** is an alternate command interpreter that you have written or obtained, then the system would load MYCOM.COM in lieu of COMMAND.COM during bootup.

The other option you can choose in CONFIG.SYS is to specify the AVAILDEV statement. For example, if you wanted the default console device to always be an external terminal you could enter

```
AVAILDEV = FALSE
```

in your CONFIG.SYS file. Now, when you load COMMAND.COM, you must specify the terminal device with the *cttydev* option. For example, entering

```
COMMAND /P \DEV\dev
```

and pressing **RETURN** would give COMMAND.COM control to your terminal, where ***dev*** is the name of the device driver for your terminal.

Error Messages

```
Bad call format
```

EXPLANATION: The parameters that passed to a device driver are invalid. If you have installed a device driver that you created or acquired yourself (using the CONFIG.SYS file), then change this driver or install a different driver. If you have not installed a device driver through the CONFIG.SYS file, then contact Technical Consultation for assistance.

```
Bad command or file name
```

EXPLANATION: The command you entered does not exist on the disk you are trying to access.

```
Bad or missing Command Interpreter
```

EXPLANATION: At the time the bootup command line was entered, the default command interpreter (COMMAND.COM) or the command interpreter specified in the CONFIG.SYS file (if any) was not stored in the root directory of the default or specified disk or partition. Copy COMMAND.COM to the root directory of this disk or partition, or specify a different command interpreter through the CONFIG.SYS file. Then, boot up again.

```
Error writing to device
```

EXPLANATION:—The device specified in *cttydev* cannot be written to.

```
EXEC failure
```

EXPLANATION: This message could be caused by any of the following error conditions:

- The executable file does not exist as specified. Specify the file by using the proper file name and, if necessary, the proper drive name and path name.
- The specified file is an .EXE file containing header information that is inconsistent with the characteristics of standard .EXE files.
- Your microcomputer has insufficient Random Access Memory (RAM). Acquire and install additional memory circuitry.
- You have specified so much buffer space (through the CONFIG.SYS file) that the remaining RAM is insufficient to execute the program. Specify less buffer space through the CONFIG.SYS file.
- You have loaded *Terminate But Stay Resident commands* (such as PSCMX80) so that the remaining RAM is insufficient to execute the program. Reset, reboot, and refrain from loading *Terminate But Stay Resident commands*.
- You have used an invalid function. Use function number 0, 1, or 3 instead.
- You have exceeded the limited amount of space allocated for the environment (a series of ASCII strings that are used by executable programs) with ASCII values set by the SET command. Redefine these values by using the SET command to use less of the environment space.

COMMAND

After performing any of these solutions, try again to execute the program that you were trying to execute when the error message occurred.

`Invalid device`

EXPLANATION: An invalid device was specified in *cttydev*.

`Invalid directory`

EXPLANATION: You tried to access or specify a directory which does not exist; double-check and reenter.

`Invalid drive specification`

EXPLANATION: You entered an invalid drive name; repeat the command with a valid drive specification.

`Invalid number of parameters`

EXPLANATION: You entered too many or too few parameters on the command line; double-check and reenter.

`Invalid parameter`

EXPLANATION: One of the parameters within your command line was invalid; reenter the statement with valid parameters.

```
Invalid path, not directory,
or directory not empty
```

EXPLANATION: The directory path name you entered is not a directory, but a file.

`Invalid path or file name`

EXPLANATION: The path name or file name you entered was invalid; double-check and reenter.

COMP (Transient)

Purpose

Compares the contents of two files or two groups of files. May be run after a COPY operation to ensure that both copies of the file(s) are identical.

NOTE: Use COMP to compare selected *files*; use DISKCOMP to compare the contents of two *disks*.

Entry Forms

```
COMP ?
COMP [filespc1 [filespc2]]
```

where ? invokes the COMP help screen display;
filespc1 is the file specification identifying the first of the two files (or groups of files) that you wish to compare; and
filespc2 is the file specification identifying the second of two files (or groups of files) that you wish to compare.

Preliminary Concepts

The COMP command enables you to compare two files to find out if their contents are identical or different. By using wildcard characters, you can also use COMP to compare two groups of selected files. The files compared may be in the same or different directories, and/or on the same or different disks. In any case, any two files that you specify for comparison must be the same size.

COMP may be invoked in two ways: *interactive entry*, wherein COMP prompts you to enter the required parameters; or *command line entry*, wherein you input the required parameters when you invoke the command. When you invoke COMP, an error message will be displayed if the files you specified for comparison are not the same size, or if a directory or file you specified is not found. (Refer to Error Messages in this section.)

During file comparison, COMP displays a message for any location in the two files that contains information that does not match. The message indicates the offset into the files of the mismatching bytes, and the contents of the bytes themselves. Both the offset and the bytes' contents are displayed in hexadecimal. The mismatch error message is displayed in the form:

```
Compare error at offset xxxxxxxx
File 1 = yy
File 2 = zz
```

where *xxxxxxxx* is the location (offset) of the mismatching information,
yy is the contents of File 1 (*filespc1*) at offset *xxxxxxxx*, and
zz is the contents of File 2 (*filespc2*) at offset *xxxxxxxx*.

This message is displayed for every mismatch found, up to a maximum of ten mismatches. If ten mismatches are found, COMP concludes that further comparison would be useless, stops the comparison operation, and displays:

```
10 Mismatches -- ending compare
```

COMP then checks whether there are any other files to compare. If additional files were specified (if wildcards were used in *filespc1* and/or *filespc2*), COMP begins the next comparison operation. If no additional files were specified, then COMP prompts you to indicate whether or not you wish to compare more files.

After successful completion of a file comparison operation (after comparing two files and finding no mismatches), COMP displays:

```
Files compare OK
```

COMP then begins the next file comparison operation (if wildcards were used) or prompts you to indicate whether or not you wish to compare more files.

In all comparison operations, COMP checks the last byte of the files being compared to make certain that each contains a valid end-of-file mark (CTRL-Z, which is hexadecimal 1A). If the end-of-file marks are found, COMP continues execution. If the end-of-file marks are not found, COMP displays:

```
Eof mark not found
```

and then continues execution.

This check is performed because some programs produce files whose sizes are always recorded in the directory as a multiple of 128 bytes, even though the actual usable data in a file is likely to be a few bytes less than the file size recorded in the directory. For files such as these, COMP may produce mismatch error messages for the last few bytes of the last 128-byte block of a file. (COMP always compares the number of bytes reflected in the files' directory entries.) Thus, the `Eof mark not found` message indicates to the user that mismatch errors displayed (if any) may not have occurred in the usable data portion of the file(s) but in the last few (empty) bytes of the file(s).

COMP does not wait for you to insert a disk containing the file(s) to be compared. Therefore, if a file to be compared is not on the same disk as the COMP command itself and if you have only one floppy disk drive in your system, you can use the interactive entry method to invoke COMP. Then, when COMP prompts you for the required parameters, you can insert the correct disk before you respond to the prompts.

COMP Help Screen

To display the COMP help screen, enter:

```
COMP ?
```

at the system prompt and press **RETURN**. The screen will display a summary of COMP usage and valid command line entry forms as shown in Figure 11.8a. The help screen is followed by the system prompt, making it easy for you to refer to the information provided as you enter a COMP command line.

```
                         COMP Version 2.xx


The COMP  utility  compares  the contents  of  two  files  or  two
specified  sets  of  files.  Normally,  COMP is used  after  a  copy
operation to verify that the files were copied correctly.  The files
that  are  specified for comparison can be on the same drive  or  on
different  drives.  They  can  also  be  in  the  same  or  different
directories.  Wildcard characters are allowed in one or both file
specifications.

Syntax: COMP ?
        COMP [filespc1 [filespc2]]
```

Figure 11.8a. COMP Help Screen

Interactive Entry Prompts and Responses

If you enter only the command name (COMP) at the system prompt and press RETURN, COMP will prompt you to enter the *filespc1* and *filespc2* parameters. This is the interactive entry method of invoking COMP. When one file comparison operation has been completed, COMP will prompt you to indicate whether you wish to compare additional files. COMP will not terminate until you indicate that you do not wish to compare any other files.

For example, if you enter:

```
COMP
```

at the system prompt and press **RETURN**, the screen displays:

```
Enter primary file name
```

You must then enter the file specification (*filespc1*) of the first of the two files you wish to compare. When you have entered the file specification, press **RETURN**. The screen displays:

```
Enter 2nd file name or drive id
```

You must then enter the file specification (*filespc2*) of the second of the two files that you wish to compare. When you have entered the file specification, press **RETURN**. The file comparison operation will begin.

When specifying the *filespc1* and *filespc2* parameters, you may use wildcard characters to compare two groups of files rather than two individual files. Using wildcard characters in both file specifications causes all of the files matching *filespc1* to be compared with the corresponding files that match *filespc2*. An example showing the use of wildcard characters is provided later in this section.

You may compare files that are in different directories and/or on different disks. If you enter only a drive name (*d*:) or path name for *filespc1* and/or *filespc2*, then COMP assumes the file name *.*. That is, COMP assumes that you wish to compare *all* files on the specified disk or in the specified directory if *.* is assumed for *filespc1*. If you enter only a drive name or path name for *filespc2*, then COMP assumes that the second file's name is the same as the first file's name.

If any mismatches are found by COMP during file comparison, an error message is displayed for each mismatch as described in Preliminary Concepts in this section. If no mismatches are found, the `Files compare OK` message is displayed. In either case, when the comparison operation has been completed, COMP prompts:

```
Compare more files (Y/N)?
```

You must enter either Y or N at this prompt. There is no default response.

If you wish to compare more files, enter **Y**. COMP will prompt you for *filespc1* and *filespc2*.

If you do not want to compare any more files, enter **N**. COMP will terminate and then the system prompt will be displayed.

Command Line Entry

The command line entry method of invoking COMP requires that you enter command line parameters for the files to be compared. That is, you must enter file specifications for *filespc1* and *filespc2*. When the file comparison operation has been completed, COMP will prompt you to indicate whether you wish to compare additional files. If you indicate that you *do* wish to compare more files, COMP begins operating in the interactive mode; that is, COMP will prompt you for the required parameters.

The specifications you enter for *filespc1* and *filespc2* may be for files on the same or different disks and/or in the same or different directories. You may use wildcard characters to compare two groups of files rather than two individual files. Using wildcard characters in both file specifications causes all of the files matching *filespc1* to be compared with the corresponding files that match *filespc2*. An example showing the use of wildcard characters is provided later in this section.

If you enter only a drive name (*d*:) or path name for *filespc1* and/or *filespc2*, then COMP assumes the file name *.*. That is, COMP assumes that you wish to compare *all* files on the specified disk or in the specified directory if *.* is assumed for *filespc1*. If you enter only a drive name or path name for *filespc2*, then COMP assumes that the second file's name is the same as the first file's name.

If you enter only one valid file specification on the COMP command line, COMP assumes that the file specified is *filespc1* and will prompt you for *filespc2*. That is, if you enter:

```
COMP filespec
```

at the system prompt and press **RETURN**, the screen will display:

```
Enter 2nd file name or drive id
```

You must then enter a valid file specification (or drive or path name, if the second file has the same name as the first file) for *filespc2* and press **RETURN**.

If any mismatches are found by COMP during file comparison, an error message is displayed for each mismatch as described in Preliminary Concepts in this section. If no mismatches are found, the `Files compare OK` message is displayed. In either case, when the comparison operation has been completed, COMP prompts:

```
Compare more files (Y/N)?
```

You must enter either Y or N at this prompt. There is no default response.

If you wish to compare more files, enter **Y**. COMP begins operating in the interactive mode and will prompt you to specify the files to be compared.

If you do not want to compare any more files, enter **N**. COMP will terminate and the system prompt will be displayed.

Comparing Two Files

Suppose you have just copied a data file from a program disk in drive A to a data disk in drive B and that you want to make sure the two files are identical before you delete the original file from the program disk. Suppose that the original file's name is OUTPUT.DAT and that you copied it to file BACKUP.DAT. To check whether the two files' contents are identical, you could enter:

```
COMP A:OUTPUT.DAT B:BACKUP.DAT
```

at the system prompt and press **RETURN**.

NOTE: This and the following examples assume that COMP is available on the disk in the default drive. If it is not, you must precede the command name with the appropriate drive name (*d*:).

When you press RETURN, comparison of the two files begins and the screen displays:

```
A:OUTPUT   .DAT and B:BACKUP   .DAT
```

If no mismatches are found (if the two files are in fact identical), the screen displays:

```
Files compare OK
```

and then COMP prompts:

```
Compare more files (Y/N)?
```

If any mismatches are found (if the files are not identical), the screen displays a message in the form:

```
Compare error at offset xxxxxxxx
File 1 = yy
File 2 = zz
```

for each mismatch (up to a maximum of ten) as described in Preliminary Concepts in this section. If mismatches were found, you would want to repeat the copy operation before deleting the original file.

In either case (whether the files are identical or not), COMP prompts:

```
Compare more files (Y/N)?
```

when the comparison operation is complete.

Using Wildcard Characters with COMP

Suppose you wish to compare two groups of files that have corresponding primary file names but different extensions. Assume that one group of files is on the disk in drive A and consists of the files LETTER.TXT, OUTLINE.TXT, and CONTRACT.TXT. Assume that the other group of files is on the disk in drive B and consists of the files LETTER.BAK, OUTLINE.BAK, and CONTRACT.BAK. To compare these two groups of files, you could enter:

```
COMP A:*.TXT B:*.BAK
```

at the system prompt and press **RETURN**. For each file on the disk in drive A that has the extension .TXT, COMP will locate the .BAK file on the disk in drive B that has the same primary file name and compare those two files.

As each pair of files is compared, the screen displays:

```
A:filename.TXT and B:filename.BAK
```

where *filename* is the primary file name of the files being compared.

For each mismatch found during each comparison, the screen displays a mismatch error message as described in Preliminary Concepts in this section.

For each comparison that is successfully completed (for each comparison in which no mismatches are found), the screen displays:

```
Files compare OK
```

before beginning the next comparison.

When all files have been compared, COMP prompts:

```
Compare more files (Y/N)?
```

Note that you may also use wildcard characters to compare one file to several files in sequence. When you do this, however, you must make certain that you enter the ambiguous file specification (the file specification that includes wildcard characters) as *filespc1* and the specific file as *filespc2*. If you enter the parameters in the opposite order, *filespc1* will only be compared to the *first* file that matches the ambiguous file specification (*filespc2*). Additionally, if you enter the ambiguous file specification *.* for *filespc2*, COMP will look for a file that has the same name as filespc1.

COMP

As an example, suppose that you have a file named DATA.EXP in the current directory of the disk in drive B and you wish to compare it with a number of other files on the disk in drive E. Suppose that the files on the disk in drive E are in subdirectory \PROGRAM\ARCHIVES and are named RUN1.EXP, RUN2.EXP, RUN3.EXP, and RUN4.EXP. To compare DATA.EXP with *each* of the files in \PROGRAM\ARCHIVES, you would enter:

```
COMP E:\PROGRAM\ARCHIVES\*.* B:DATA.EXP
```

at the system prompt and press **RETURN**. The screen would display:

```
E:\PROGRAM\ARCHIVES\RUNn.EXP and B:DATA.EXP
```

where *n* is a number from 1 through 4.

If any mismatches are found, a mismatch error message is displayed for each as described in Preliminary Concepts. If no mismatches are found, the `Files compare OK` message is displayed and then the next file in \PROGRAM\ARCHIVES is compared with DATA.EXP. The screen displays:

```
E:\PROGRAM\ARCHIVES\RUNn.EXP B:DATA.EXP
```

for the new file comparison. File comparison continues in this fashion until all four files in \PROGRAM\ARCHIVES have been compared with DATA.EXP.

When the comparisons are completed, COMP prompts:

```
Compare more files (Y/N)?
```

If you had entered the command line parameters in the opposite order (if you had entered COMP B:DATA.EXP E:\PROGRAM\ARCHIVES\*.*), COMP would have attempted only one file comparison. That is, COMP would have looked for a file named DATA.EXP in the specified directory and, if found, compared it to DATA.EXP in the current directory of the disk in drive B.

You can also invoke COMP using assumed wildcards. That is, if you enter only a drive name or path name for a file specification, COMP assumes the filename *.*. An example of when you might invoke COMP in this way is if you wished to compare all files in one directory with all files in another directory. For example, suppose you wish to compare files in two subdirectories on the default disk and that the subdirectories are \THIS\DIR and THAT\SUBDIR. To compare all the files in these two subdirectories, you could enter:

```
COMP \THIS\DIR \THAT\SUBDIR
```

at the system prompt and press **RETURN**. COMP would complete file comparisons as if you had entered the command line:

```
COMP \THIS\DIR\*.* \THAT\SUBDIR\*.*
```

Error Messages

```
Files are different sizes
Compare more files (Y/N)?
```

EXPLANATION: COMP will only compare files that are the same size, as reflected in the files' directory entries. This message is displayed if the files you specified for comparison are not the same size. If you wish to compare other files, enter **Y** at the prompt and continue with COMP. If you do not want to compare any other files, enter **N**. The system prompt will be displayed.

```
filespec - File not found
Compare more files (Y/N)?
```

EXPLANATION: This message is displayed if a file you specified (for either *filespc1* or *filespc2*) is not on the default or specified disk or in the current or specified directory. Make sure that you entered the correct file specification and that the appropriate disk is available. To invoke the desired file comparison, enter **Y** at the prompt and then enter the correct file specifications at the COMP prompts.

```
Incorrect DOS version
```

EXPLANATION: This message will be displayed if you booted up your system with a version of MS-DOS previous to version 2. In order for you to use COMP, MS-DOS version 2 or higher must be resident in memory.

```
Insufficient memory
```

EXPLANATION: This message is displayed if available memory is not sufficient for COMP to complete file comparison. If this occurs, you may do one of the following to free up memory:

- Reboot your system to clear memory of programs that remain resident in memory after execution. (Such programs, called terminate-and-stay-resident programs, include the PSC utilities.) Reinvoke COMP before you run any other programs.

- If simply rebooting your system does not solve the problem, you can reduce the number of BUFFERS defined in the CONFIG.SYS file. (For information on the CONFIG.SYS file, refer to Chapter 9, "System Component Features.") After reducing the number of buffers, reboot your system and reinvoke COMP.

```
Invalid drive specification
```

EXPLANATION: This message is displayed if you entered an invalid drive name as part of the COMP command line or in response to a COMP prompt. This message is followed by the system prompt or by a COMP prompt. Reenter the drive name or file specification, making sure that you use a drive name that is valid for your system.

```
pathname - Invalid path
Compare more files (Y/N)?
```

EXPLANATION: This message is displayed if a directory path name you entered either is not a valid path name or does not exist on the default or specified disk. If you wish to continue with COMP, enter **Y** at the prompt and respond to the prompts with valid entries. If you do not wish to continue with COMP, enter **N** at the prompt. The system prompt will be displayed.

CONFIGUR (Transient)

Purpose

The CONFIGUR program configures the system to use the correct protocol for your printer, modem, or other serial/parallel device. It also includes an option that allows you to assign your MS-DOS partitions manually or automatically.

Entry Form

```
CONFIGUR
```

Preliminary Concepts

CONFIGUR is a program that allows you to configure your system's logical devices so that they fit the protocol requirements of your printer, modem, or other serial/parallel devices that you connect to your microcomputer. *Protocol* is the word used when referring to the method of communication required for data transfer between a microcomputer and a peripheral. The protocol also offers a means of controlling the manner in which the peripheral performs its specific operations. A device's protocol consists of a set of standard signals that the hardware device recognizes and uses so that its operation can be correctly synchronized and interpreted.

NOTE: If you are using only a parallel printer, connect it to the parallel port. This port is already configured for LPT1, so you do not need to run the CONFIGUR program.

CONFIGUR

CONFIGUR offers you a quick way to select the protocol that your devices can recognize. CONFIGUR enables the Input/Output Manager (IO.SYS) to communicate with your devices by using the correct baud rate, communication lines, and control and data signals necessary for information transfer to a particular peripheral device.

Since CONFIGUR adjusts your system for hardware characteristics, you must answer prompts that ask about these characteristics. Always consult your hardware manuals and check the settings of your hardware devices when answering these prompts.

CONFIGUR also contains an option that allows you to either use the ASSIGN command to assign MS-DOS partitions manually or have the partitions automatically assigned when you boot up. CONFIGUR sets a flag contained in IO.SYS, which causes either automatic or manual partition assignment.

CONFIGUR is a *menu-driven* command. This signifies that your screen will display a list, or menu, of selections from which you can choose. The first menu, referred to as the main menu, contains choices that will lead you to a series of submenus. Many of these submenus will list protocol characteristics for peripheral devices. Refer to the manual for a specific peripheral when selecting values from these menus to be certain you are selecting the proper values.

To use the menu, press the letter that corresponds to the menu selection you want to choose.

The configurable logical devices are the PRN (LPT1, 2, and 3) and AUX (COM1 and 2) devices. A logical device name is assigned to a table of information that is contained in IO.SYS. When that device name is used in a command to read or write data, IO.SYS uses the information it has about that device to execute the command.

These devices are named to the system as shown in Table 11.1.

Logical devices configurable to different identities make it easy for the operating system to use many devices. If the operating system could not use a logical device to represent a physical device, it would either have to be rewritten every time a different device was used or contain information about all possible devices.

Logical devices present a very practical approach to enable the operating system to handle many different devices. When confronted with one of the reserved logical device names (for example, LPT1 or COM1), the operating system knows exactly where to look to get information currently used to describe the device.

Table 11.1. System Device Names

DEVICE NAME	DEVICE FUNCTION
PRN or LPT1	Parallel printer device #1
AUX or COM1	Serial device #1
COM2	Serial device #2
LPT2	Parallel printer device #2
LPT3	Parallel printer device #3

CONFIGUR also knows exactly where the descriptions of the logical devices are stored. When CONFIGUR is told to save a new description, it overwrites the old device description. Therefore, if the PRN device is currently the description of an MX-80 printer, commands that have PRN as an argument cause the operating system to use the MX80 description. If CONFIGUR is then invoked and the PRN device is redefined to the characteristic protocol of a Diablo 1640 printer, the operating system then recognizes and uses the different protocol. Even though the actual physical device changes, MS-DOS is able to communicate to it because it looks up the device's definition under the logical name.

The following discussion of CONFIGUR is divided into five areas, each describing the configuration of a particular device or the automatic/manual assignment of Winchester partitions.

The areas covered include:

1. the main menu. This is a description and explanation of the main menu in CONFIGUR and the significance of the various selections.

2. configuring an LPT device. This is a printer device that uses one of the parallel ports.

   NOTE: If you are using a serial printer, follow the instructions given in Remapping an LPT Device.

3. configuring a COM device. This is a serial device that includes such peripherals as modems and some printers.

4. remapping an LPT device. This is typically used to configure your system for a serial printer, such as the MX80-Serial. To accomplish this, you must use the following procedure:

   a. Configure a COM device (see 3 above) for your particular printer.

   b. Remap LPT1 to the COM device you configured for your printer. Make sure the COM devices are the same. For example, if you configure COM1 for your printer, you must remap LPT1 to COM1, not COM2.

   c. Use a *null modem* cable to connect your printer to the configured COM port.

5. changing the automatic partition assignment flag. This allows you to choose whether your Winchester partitions are manually or automatically assigned each time you boot up your system.

The Main Menu

To begin to configure your system for a particular device's protocol and/or to choose how your partitions are assigned upon bootup, enter:

```
CONFIGUR
```

and press **RETURN.** CONFIGUR will display a menu of choices for configuring your system. Since CONFIGUR checks your hardware configuration when it is invoked, it knows which ports are connected to your system and will only display valid choices. For example, if you did not have any serial or parallel ports connected to your microcomputer and you invoked CONFIGUR, it would display:

```
No serial or parallel devices are attached.
No configuration is possible.
```

Now, examine the menu that would appear if you have a standard configuration (a system with one parallel port and two serial ports). In this situation, when you enter:

```
CONFIGUR
```

at the system prompt and press **RETURN**, the screen will display:

```
               CONFIGUR Version 2.x

Use one of the following options to configure a device

A. Configure LPT device
B. Configure COM device
C. Change automatic partition assignment flag

D. Exit with no changes

Enter selection (A-D)
```

At this main menu, select either A, B, C, or D.

NOTE: The MS-DOS version numbers are shown in the screen displays as the variable, 2.*x*. This is because your screen display may be from a more recent version than that available when this manual was prepared.

Pressing D returns you to the system prompt without making any configuration changes.

Depending on the selection you choose from this main menu, additional menus may appear that offer alternate system configuration options.

Because there are several menus and options that are available (depending on the options you select), ending the program is discussed first before describing the various configurations that are available.

After making your selections, you will return to this main menu (notice that three additional options appear):

```
                CONFIGUR Version 2.x

Use one of the following options to configure a device

A. Configure LPT device
B. Configure COM device
C. Change automatic partition assignment flag

Use one of the following to modify an existing system

D. Exit program
E. Make changes to disk
F. Make changes to memory
G. Make changes to both disk and memory

Enter selection (A-G):
```

Options E, F, and G will appear on the main menu after you have made a configuration. These options allow you to save the configuration that you have selected.

You can still exit the program without saving any of the options that you have selected by pressing D.

Option E records on disk the changes that you selected. After selecting option E, you are asked on which disk you want the changes recorded:

```
Enter drive name with system to modify (A-x):
```

CONFIGUR

You can record the configuration you selected to any valid drive in your system, hence the *x*. The *x* represents the disk drive having the greatest alphabetic drive name in your particular system.

This feature makes it possible to configure the system on disks other than your current system disk. This can be a useful feature if you have other disks that will be used with different devices for special purposes. This means you can change your current system disk's configuration without disturbing the system configuration that is currently in memory at the time you use CONFIGUR.

Selecting option F alters the system configuration that is currently in your microcomputer's memory without making that configuration permanent by saving it to a disk. When you press CTRL-ALT-DEL and reboot the system, the configuration you selected will be erased and the one on the disk will be read into memory again. This allows you to make a temporary change.

NOTE: Changes specified through the CONFIGUR utility can only be recorded on a disk if the disk is *write-enabled*. If you are performing the CONFIGUR operation on a *write-protected* disk, you can only apply changes to the system in memory.

Select option G if you want to make the changes to both memory and disk. This option also asks on which drive's disk you want to record the changes. Therefore, you can change memory and disks other than your booted disk, if you want.

Options A, B, and C lead to sets of secondary menus that look similar to the main menu.

Configuring an LPT Device

If you select option A, Configure LPT device, from the main menu, the following submenu will appear:

```
A. Map parallel output to serial output
B. Configure parallel device

C. Exit

Enter selection (A-C):
```

If you want to configure a parallel device, enter option B to configure the parallel device. (Mapping parallel output to serial output is covered in a later section.) Selecting option B will display:

```
Select the parallel device to be configured

A. LPT1

B. Exit

Enter selection (A-B):
```

NOTE: In this example there is one LPT device, LPT1. Your system may have one, two, or three LPT devices. The menu will reflect your particular system.

If you choose to configure the LPT1 parallel device, by entering option **A**, the following prompts will appear sequentially:

```
Answer the following questions with Y for Yes and N for No

Strip parity on output? (Y/N) <N>
Map lower case to upper case on output? (Y/N) <N>
```

NOTE: In the previous questions, the answers in angle brackets (<>) are the defaults and can be entered by pressing RETURN.

If you wish to strip parity on output (set the eighth bit of each character being sent to the LPT device to 0), press **Y**. If you wish all lowercase characters to appear as uppercase, press Y. After you have answered the questions, the following series of questions will be displayed:

```
If you do not wish a pad character, simply
 press the RETURN key, and then enter a zero
 as the number of pad characters, otherwise press
 the actual key character you wish to pad

For example, to pad after all carriage returns, press
 the RETURN key.

Press the key corresponding to your desired pad character:
Enter the number of pad characters to send (0-255):
```

NOTE: The option to insert pad characters is provided for older, and hence, slower, printer devices which may not be able to react as quickly to the signals coming from your microcomputer. Usually pad characters are sent after a RETURN to allow time for the print element in your printer to return to column 1. It may need this extra time if it is of the older variety.

```
The timeout value is used to give slow devices time to respond to
Input/Output requests. A small value is usually sufficient, but a
number from 0 to 255 can be entered.


Enter timeout value for LPT1:
```

NOTE: The timeout value is provided for reasons roughly similar to the pad characters. However, a timeout value can be set for any type of device, not just printers. The timeout value is a number from 0 to 255 that represents the number of "cycles" that your microcomputer should wait for a device to respond, before it stops trying to communicate. The slower the device you are configuring your microcomputer for, the larger the number you should enter for the timeout value.

After you have entered a timeout value (even if it is zero), you will be returned to the following menu:

```
Use one of the following options to select the type of configuration

A. Map parallel output to serial output
B. Configure parallel device

C. Exit

Enter selection (A-C):
```

If you have configured everything that you need, press C and exit. This will return you to the main menu shown again on the next page:

```
              CONFIGUR Version 2.x

Use one of the following options to configure a device

A. Configure LPT device
B. Configure COM device
C. Change automatic partition assignment flag

Use one of the following to modify an existing system

D. Exit program
E. Make changes to disk
F. Make changes to memory
G. Make changes to both disk and memory

Enter selection (A-G):
```

If you have completed the necessary configuration for your system, save the changes you have made to either disk, memory, or both.

If you also need to configure a COM device, do not save the changes you have already made at this time, but select option B, Configure COM device.

Configuring a COM Device

If you select option B, Configure COM device, from the main menu, the following submenu will appear:

```
Select the serial port to be configured

A. COM1
B. COM2

C. Exit

Enter selection (A-C):
```

NOTE: Two COM devices are shown in this example. If your system has only one COM device, only COM1 will be displayed in this menu.

If you choose option **A**, COM1, the following menu will be displayed:

```
Use one of the following options to select the appropriate configuration


A. Compatibility mode (2400 baud, DTR and RTS pos.)
B. MX-80 (4800 baud, DTR pos. (pin 20))
C. H/Z-25 (4800 baud, RTS pos. (pin 4))
D. H-14/WH-24 (4800 baud, RTS Neg. (pin4))
E. Diablo 630/1640 (1200 baud, ETX/ACK)
F. WH-23/WH-33/WH-43 modem (300 baud, No handshake)
G. WH-12 Votrax Type-N-Talk (4800 baud, RTS Pos. (pin 4))
H. User Defined
I. Exit with no changes

Enter selection (A-I):
```

If you wish to configure your system for one of the devices shown, you need only enter the letter of your choice. The system will be configured for that device, and you will automatically be placed back in the main menu. The devices named cover a broad range of printers, modems, and special devices. There is even a special compatibility mode that provides IBM type compatible handshaking for serial devices (selection A). However, if you wish to configure your system for a serial device that is not listed in the above menu, you will have to choose selection H, User Defined.

To illustrate how this works, choose option **H** and the following prompts will be displayed in sequence:

```
Answer the following questions with Y for Yes and N for No

Strip parity on input? (Y/N) <N>
Strip parity on output? (Y/N) <N>
Map lowercase to uppercase on input? (Y/N) <N>
Map lowercase to uppercase on output? (Y/N) <N>
```

NOTE: In the previous questions, the answers in angle brackets (<>) are the defaults and can be entered by pressing **RETURN**.

The first two prompts in this sequence ask if you want the parity (error checking) information bits to be removed from the data on either input or output.

The next two questions ask if you wish to have all lowercase characters read as uppercase on input, or printed as uppercase on output.

After you have answered the previous questions, the following menu will be displayed. This menu requests that you enter the speed at which your microcomputer will communicate (baud rate) with the peripheral device for which you are configuring the system. As before, consult your peripheral's documentation for information on baud rate.

```
Select one of the following baud rates

A. 110
B. 150
C. 300
D. 600
E. 1200
F. 2400
G. 4800
H. 9600

Enter one of the baud rate values:
```

After you have answered the questions and entered the appropriate baud rate for the device you are using, the following series of menus and prompts will be displayed:

```
Use one of the following stop bit values

A. 1 Stop bit
B. 2 Stop bits

Enter one of the stop bit values:
```

The stop bits are set to tell the peripheral device when the data communicated from your microcomputer has ended. Again, consult your peripheral device documentation for this information.

```
Use one of the following parity selections

A. No parity
B. Odd parity
C. Even parity

Enter one of the parity values:
```

Parity is a technique used by your microcomputer to check for errors in the data communication between the microcomputer and peripherals.

```
Use one of the following to select the word length
NOTE: Word length is exclusive of stop bits and parity

A. 7 bit words
B. 8 bit words

Enter one of the word length values:
```

The word length is the size of the data that are communicated between your microcomputer and a peripheral.

The following submenu asks you to enter the type of special communications protocol or *handshaking* required by your particular peripheral device. You should find this information in the documentation for the device in question. This menu is described in three separate sections, based on the type of handshake (or absence of handshake).

```
Use the following to select a handshaking protocol

A. No Handshaking
B. ETX/ACK
C. DC1/DC3
D. Compatibility Mode, DTR and RTS Positive
E. RTS Positive (pin 4)
F. RTS Negative (pin 4)
G. DTR Positive (pin 20)
H. DTR Negative (pin 20)

Enter one of the handshake values:
```

The first group contains only the first option, A (No Handshaking). This option is for devices that do not need handshaking and will not require timeout values to be entered, although you may choose to insert pad characters.

The second group is referred to as *software handshake*. This group includes options B (ETX/ACK) and C (DC1/DC3). These options will allow you to insert pad characters but will not require timeout values to be entered. In addition, a special prompt will appear if you select B (ETX/ACK):

```
Number of characters between the ETX/ACK handshake (0-255):
```

NOTE: This prompt will only appear if item B was selected from the menu.

The third group is referred to as *hardware handshake*. This group includes selections D through H:

```
D. Compatibility Handshaking (DTR and RTS pos.)
E. RTS Positive (pin 4)
F. RTS Negative (pin 4)
G. DTR Positive (pin 20)
H. DTR Negative (pin 20)
```

These options will allow you to select both a timeout value and a pad character.

The screen prompts for pad characters and timeout value are described below:

```
If you do not wish a pad character, simply
 press the RETURN key, and then enter a zero
 as the number of pad characters, otherwise press
 the actual key character you wish to pad

For example, to pad after all carriage returns, press
 the RETURN key.

Press the key corresponding to your desired pad character:
Enter the number of pad characters to send (0-255):
```

NOTE: The option to insert pad characters is provided for older, and hence, slower, printer devices which may not be able to react as quickly to the signals coming from your microcomputer. Usually pad characters are sent after a RETURN to allow time for the print element in your printer to return to column 1. It may need this extra time if it is of the older variety.

NOTE: The following prompt will only be displayed if option D, E, F, G, or H was selected from the handshake menu.

```
The timeout value is used to give slow devices time to respond to
Input/Output requests. A small value is usually sufficient, but a
number from 0 to 255 can be entered.

Enter timeout value for COM1:
```

NOTE: The timeout value is provided for reasons roughly similar to the pad characters. However, a timeout value can be set for any type of device, not just printers. The timeout value is a number from 0 to 255 that represents the number of "cycles" that your microcomputer should wait for a device to respond, before it stops trying to communicate. The slower the device you are configuring your microcomputer for, the larger the number you should enter for the timeout value.

Enter a timeout value, even if it is zero.

After you have finished entering the answers to these prompts, you will automatically be returned to the main menu. When the main menu displays, you will have the choice of exiting the program without making any changes; or making changes to disk, memory, or both.

Remapping an LPT Device

If you wish to use a printer with a serial interface you not only need to configure a COM device for that printer, but you need to logically connect, or remap the parallel output to the serial output of that COM device. This step is absolutely necessary if you plan to use a serial printer with your microcomputer. The procedure for remapping is described below. To begin, enter

```
A>CONFIGUR
```

and press **RETURN**; the screen will display

```
                CONFIGUR Version 2.x

Use one of the following options to configure a device

A. Configure LPT device
B. Configure COM device
C. Change automatic partition assignment flag

D. Exit with no changes

Enter selection (A-D)
```

At this main menu press A, and the LPT menu will display:

```
Use one of the following options to select the type of configuration

A. Map parallel output to serial output
B. Configure parallel device

C. Exit

Enter selection (A-C):
```

Select option A from this menu.

NOTE: If you are going to use a serial printer, you must connect it to the COM1 or COM2 port with a *null modem* cable, available through your dealer.

After A is entered, the following submenu is displayed:

```
Select the parallel port to be mapped

A. LPT1
B. LPT2
C. LPT3

D. Exit

Enter selection (A-D):
```

NOTE: Your system may have only one or two parallel ports, however, this menu will always be shown with all three LPT devices. This is because they represent logical devices, as opposed to physical devices, and can always be remapped.

If you choose **A**, the following menu is displayed:

```
Use one of the following to select a mapping for LPT1

A. No mapping
B. Map to COM1
C. Map to COM2

Enter one of the mapping values:
```

Choose option B or C, depending on which COM port you want the LPT device to be mapped—that is, to which COM port you wish the printer connected.

Option A (No Mapping) is used to undo a previous remapping or *unmap*. This returns the system to the default values.

Whatever option you select at this point will place you back in the menu two levels previous to the last one, as shown:

```
Use one of the following options to select the type of configuration

A. Map parallel output to serial output
B. Configure parallel device

C. Exit

Enter selection (A-C):
```

If you have configured everything that you need, press **C**, and exit. This will bring you back to the main menu shown again on the next page:

CONFIGUR

```
                CONFIGUR Version 2.x

Use one of the following options to configure a device

A. Configure LPT device
B. Configure COM device
C. Change automatic partition assignment flag

Use one of the following to modify an existing system

D. Exit program
E. Make changes to disk
F. Make changes to memory
G. Make changes to both disk and memory

Enter selection (A-G):
```

CAUTION: If you have not previously configured the COM device to which you mapped the parallel output, do so now. When you use a serial printer, configure the COM port to be used (either COM1 or COM2) and remap the parallel output to that port. If you do not complete both of these steps, your serial printer will not function.

If everything is complete, save the changes you have made to either disk, memory, or both.

Changing the Automatic Partition Assignment Flag

If you select option C, Change automatic partition assignment flag, from the main menu, the following submenu will appear:

```
The system is currently set for automatic partition assignment.

Use one of the following to select assignment mode.

A. Automatic partition assignment.
B. Manual partition assignment.

C. No change, exit to main menu.

Enter selection (A-C):
```

The first line in this submenu indicates how partition assignment is currently set. The preceding submenu indicates that the system is currently set for automatic partition assignment. However, your submenu might display *manual assignment*, depending on how your system is currently configured.

If you want to configure your system to automatically assign all MSDOS partitions on your Winchester to the next available drive letter(s), select option A.

NOTE: The Topview and Microsoft Project application programs check all available drives upon startup. Therefore, you *must* configure your operating system for automatic partition assignment if you are using either of these programs.

If you want to configure MS-DOS so that partitions must be manually assigned a drive letter each time you boot up, then select option B. Remember, when booting up from a floppy disk, each Winchester partition must be assigned a drive letter before it can be accessed. If you boot up from an MS-DOS partition, that partition is automatically assigned the first available drive. However, all other MS-DOS partitions on your Winchester must be assigned a drive letter before they can be accessed. (Refer to the ASSIGN command in this chapter for details on assigning a drive letter to an MS-DOS partition.)

In order for the change in the automatic partition assignment to be implemented, you must reboot your system. Therefore, once all the changes to CONFIGUR are complete, you must record the changes on disk. Changes recorded only in memory are lost when you reboot.

If you decide to leave the partition assignment flag as it is currently set, select option C to return to the main menu.

Advanced Concepts

The CONFIGUR program provides a way for you to change peripherals as your needs change. If you need to print quickly, you might choose a high-speed dot-matrix printer, which might use an RTS positive handshake of 4800 baud. Or, your printer needs might require a cleaner, crisper look for correspondence and contracts—so, you might choose a daisywheel printer that uses ETX/ACK handshaking and operates at 1200 baud.

CONFIGUR makes it possible for you to connect both printers at the same time or alter the configuration quickly so that you are able to use one printer, run CONFIGUR, connect a different printer, and immediately start to use *that* printer.

Certain applications require a modem. CONFIGUR sets your system configuration so that you are able to use a modem easily.

Disks that contain applications for specialized use can be quickly modified with different system configurations that are changed to memory only, without changing the configuration of the system on your disk.

Error Messages

```
Cannot locate file IO.SYS
Press any key to continue...
```

EXPLANATION: The disk you are trying to configure does not contain IO.SYS.

```
IO.SYS file version incorrect
Press any key to continue...
```

EXPLANATION: The version of IO.SYS contained on disk is incompatible with the one CONFIGUR is set up for.

```
No serial or parallel devices are attached.
No configuration is possible.
```

EXPLANATION: Your system must have at least one parallel or one serial port connected in order to run CONFIGUR.

```
Version mismatch with IO.SYS
```

EXPLANATION: The version of IO.SYS in memory is incompatible with the one CONFIGUR is set up for.

COPY (Resident)

Purpose

Use this command to copy one or more files from one location (source) to another (destination) that may or may not be on the same disk or in the same directory.

Entry Forms

```
COPY filespec [d:]filename[/V]
COPY filespec d:[/V]
COPY filespec [d:]pathname[/V]
COPY [d:]pathname[d:][pathname][/V]
```

where ***filespec*** is the file specification of the source file(s) you wish to copy;
d is the drive name of the source or destination drive;
filename is the file specification of the destination file (the file you want the first file copied to);
pathname is the path name identifying the directory, if other than the source directory, to which you want the file copied; the path name may or may not include a file name; and
/V is the Verify switch.

Command Line Entry

Below are described the parameters of the COPY command line. Some parameters are required and others are optional. Entry requirements are noted where appropriate in the parameter descriptions.

Source File Specification

The source *filespec* must always be specified as the first parameter following the COPY command. It is this file that the system will copy to the destination you specify. The source may be a full file specification or only a primary file name and extension (if one exists). If you do not specify a drive name as part of the source, the system will search the default drive to locate the source file. The COPY command has no effect on the source file; the system simply reads the file contents and writes them to the specified destination.

If the source file is on a disk other than the default disk or in a directory other than the current directory, you must enter the drive name (*d*) or directory path (*pathname*) as part of the source file specification.

You may use wildcard characters in the source *filespec* when combining multiple files in a single destination or when copying more than one file to the destination. Refer to Concatenating Files in this section.

NOTE: If you enter a source file specification that consists of a pathname that does *not* include a filename, it is equivalent to specifying [*d*:] *pathname\.*. That is *all* files in the specified directory will be copied.

Destination File Specification

You may specify the destination to which the source file is to be copied in a number of ways, each of which produces a unique result. The parameters that you may use in specifying the destination are:

- destination drive name (*d*)
- destination path name (*pathname*)
- destination file specification (*filespec*)

COPY

If the destination is a drive name (*d*), the source file will be copied to the current directory of the drive specified, and the resulting file will be given the same name as the source file.

If the destination is a full file specification (drive name, primary file name, and extension), the source file will be copied to the specified drive, to a file with the primary file name and extension specified.

If the destination is a file name, the source file will be copied to the source (or default, if the source is not the default) drive under the file name specified. That is, two files with identical contents and different file names will exist on the source disk when the copy operation is completed.

If the destination is a drive name (*d*) and path name (*pathname*), the source file will be copied to the specified directory (and file, if a file name is entered as part of path name) on the destination drive. If no destination file name is entered, the destination file will have the same file name as the source file.

If the destination is a *pathname* only, the source file will be copied to the directory (on the default disk) specified. If a file name is entered as part of the path name, the destination file will have that name. If no destination file name is entered, the destination file will have the same file name as the source file.

NOTE: Refer to Chapter 7, "Directory Features," for more information on path names.

You may use wildcard characters in the destination file specification when concatenating files. Refer to Concatenating Files in this section.

In specifying source and destination files, be aware that a file cannot be copied to itself. That is, if the source file specification references a file on the default disk and a destination file specification is not entered, the copy operation will be aborted and the screen will display

```
File cannot be copied onto itself
       0 File(s) copied
```

The system prompt appears following this message.

At the end of a completed copy operation, the screen normally displays

```
n File(s) copied
```

where *n* is the number of destination files created.

Verify Switch

The Verify switch (/V) used with the COPY command causes MS-DOS to verify that the sectors written on the destination disk during the copy operation are properly recorded. Recording errors rarely occur during execution of the COPY command; however, this switch is provided as an option to enable you to be doubly sure that critical data has been accurately transferred.

If you use the Verify switch, be aware that this option causes the COPY command to be executed more slowly than it would be otherwise, as MS-DOS must check each entry recorded on the destination.

Copying a File to Another Disk

Suppose you have file DATA.SPL on your default disk and want to transfer it to another disk for storage. Assuming that drive A is the default, you could copy the file to the disk in drive B by entering

```
COPY DATA.SPL B:
```

at the system prompt and pressing **RETURN**.

The system copies the file, under the same file name, to the current directory of the disk in drive B. When the copy operation is complete, the screen displays

```
1 File(s) copied
```

followed by the system prompt.

Copying and Renaming Files

Suppose you want to copy file DATA.SPL from a data disk to your current system default disk for use in the current work session, and wish to rename the file for its current use. If the file is on the disk in drive B and the default disk is in drive A, you can copy and rename the file by entering

```
COPY B:DATA.SPL A:SPLDATA.RUN
```

at the system prompt and pressing **RETURN**.

The system copies file B:DATA.SPL to the disk in drive A under the name SPLDATA.RUN. When the copy operation is complete, the screen displays

```
1 File(s) copied
```

followed by the system prompt.

If you wish to copy a file to the same directory on the source disk in which the file currently resides, you must rename the file (that is, specify a unique destination file name) when you invoke the COPY command. Suppose you have file STUFF on the default disk and want to duplicate the file in the same directory level of the disk. You cannot copy a file to itself, but you can copy a file to the same disk and directory if you rename it, resulting in two files with identical contents and different file names. To do this, you could enter

```
COPY STUFF STUFF2
```

and press **RETURN**.

The system copies the contents of file STUFF to a file named STUFF2, and then displays

```
1 File(s) copied
```

followed by the system prompt.

Copying a File to Another Directory

You can copy a file to a different directory on the same disk without renaming the file. (See Chapter 7 for information on the MS-DOS directory structure.) If file STUFF contains information that will be used often in more than one directory, you might copy the file from the source directory to another directory on the same disk by entering

```
COPY STUFF \USER\JACK
```

and pressing **RETURN**. (In this example, USER and JACK are subdirectory names; USER is the parent directory of JACK.)

The system copies the file STUFF from the current working directory to the directory named JACK, and stores it under the file name STUFF.

NOTE: The above example assumes that the directories USER and JACK already exist. If USER exists and JACK does not, the source file will be copied to a file named JACK in the directory named USER. If neither directory exists, the copy operation will not be executed and the screen will display

```
Invalid directory
```

followed by the system prompt.

If you wish, you can copy a file to another directory and rename it. To do this for the above example in which the directories USER and JACK are assumed to exist, you could enter

```
COPY STUFF \USER\JACK\DATA
```

and press **RETURN**. In this case, the destination file would be named DATA.

Concatenating Files

The COPY command enables you to link a series of files together to create a destination file that consists of the contents of more than one source file. This linking process is known as "concatenation." To invoke concatenation, a series of file specifications rather than a single file is entered as the source *filespec*. All file specifications in the series must be separated from each other by a plus sign (+). For example, assume that the default disk was in drive A and you entered the following command at the system prompt:

```
COPY A.XYZ+B.COM+B:C.TXT BIGFILE.LRG
```

When you pressed RETURN to begin command execution, the system would copy and combine the contents of A:A.XYZ, A:B.COM, and B:C.TXT in destination file BIGFILE.LRG on the default disk. Notice that there are no spaces in the series of source files in the above example, only plus signs between the source specifications.

As the COPY command is executed, the screen displays the file name of each file copied. For the above example, the screen would display

```
A:A.XYZ
A:B.COM
B:C.TXT
      1 File(s) copied
```

Note that in this operation, the number of files copied (as shown in the screen display `1 File(s) copied`) is the number of destination files produced in the copy operation.

You may also use wildcard characters when concatenating files. (Refer to Chapter 1 for information on wildcard characters.) As an example, enter

```
COPY *.LST COMBIN.PRN
```

This command copies all files on the default drive with the extension .LST and combines them in the destination file (also on the default drive) COMBIN.PRN.

If a wildcard character is used in the destination file specification as well as in the source file specifications, it is possible for the system to perform several individual concatenations when you enter only one COPY command line. For example, suppose you entered

```
COPY *.LST+*.REF *.PRN
```

In this example, for each file matching *.LST, that file is combined with the corresponding .REF file, then written to a destination file with the same primary file name and the extension .PRN. That is, a file named FILE1.LST would be combined with FILE1.REF, and the combined results would be written to FILE1.PRN; file XYZ.LST would be combined with file XYZ.REF and written to file XYZ.PRN, etc. Notice the difference between the above command and the example that follows.

Suppose you entered

```
COPY *.LST+*.REF COMBIN.PRN
```

and pressed **RETURN**. This command causes the system to combine all files matching *.LST with all files matching *.REF and write the results to the single destination file COMBIN.PRN.

When using wildcard characters to concatenate files, be careful not to enter a COPY command where one of the source file names has the same extension as the destination. This can result in the loss of source files. For example, if you have the file ALL.LST on your default disk, the following command is an erroneous use of wildcard characters in the COPY command:

```
COPY *.LST ALL.LST
```

The error, however, will not be detected by the system until it attempts to read the contents of the file ALL.LST, and, at that point, the original file already may have been overwritten by other source file contents. The COPY program normally compares each source file specification with the destination file specification as each source is located. If the source is the same as the destination, the file is skipped and the screen displays

```
Content of destination lost before copy
```

The system goes on to locate the next source file specification, and further concatenation proceeds normally. All source files except the source that was the same as the destination will be unaffected, as is normal.

The correct way to concatenate files by using wildcard characters when one of the source files has the same extension as the destination is shown in the example below:

```
COPY ALL.LST+*.LST
```

This command causes the system to append all *.LST files except ALL.LST to ALL.LST. This command will not produce an error message. All source files except ALL.LST will be unaffected by the end result. The contents of the original file ALL.LST will not be altered or lost, but will become the first block of data contained in the destination file ALL.LST. This is because when the COPY program sees a match between the source file and the default destination file (ALL.LST, since it is the first source file specification), the source file is skipped and information from the remaining source files is simply appended to it as the destination file.

Copying a File to or from a Device

You may use reserved device names with the COPY command in lieu of file specifications to copy a file to or from a device such as microcomputer terminal. (See Chapter 1 for information on reserved device names.) For example, to copy a file to the screen for your review, you could enter

```
COPY DATAFILE CON
```

and press **RETURN**. The contents of file DATAFILE on the default disk would be displayed. To temporarily stop the display so that you can more easily read DATAFILE contents, press **CTRL-NUM LCK**. To resume the display, press **RETURN**.

NOTE: You could also display the file one screen at a time by using the above example COPY command line with the MORE command. For information on MORE, refer to the appropriate section of this chapter.

If you wished to store the contents of a screen display into a file, you could enter

```
COPY CON INPUT.DAT
```

and press **RETURN**. This command must be entered *before* you enter the information that you wish to save in file INPUT.DAT. After you enter the command, you may enter the information on the keyboard line by line, pressing RETURN at the end of each line. When you have entered all the information you wish to have copied from the terminal to the specified file, enter **CTRL-Z** and press **RETURN** to indicate end-of-file. This ends the copy operation and causes the system to save the keyboard input to file INPUT.DAT on the default disk.

Error Messages

```
Content of destination lost before copy
```

EXPLANATION: When concatenating files, you have incorrectly specified a source file that is the same as the destination file, and the contents of the source file have been lost. Concatenation of remaining files will proceed normally.

```
File cannot be copied onto itself
       0 File(s) copied
```

EXPLANATION: You have specified a destination file that is exactly the same as the source file (either by your input or use of default destination parameters). Reenter the COPY command and specify an alternate destination file (use a different drive name, file name, and/or directory). A file can be copied to the same directory on the same disk only if a unique destination file name is entered.

```
File creation error
```

EXPLANATION: This error message will be displayed under two circumstances. The first is if you attempt to copy a file to a destination file specification that is a .SYS or system "hidden" file, or that is a read-only file. The other is if you attempt to copy a file to a root directory that is full and thus does not have sufficient space remaining to contain the directory information required for the file.

When this message occurs, reenter the COPY command, but specify an alternate destination file specification or an alternate destination directory.

```
filename File not found
           0 File(s) copied
```

EXPLANATION: The source file specification you entered does not exist in the directory on the specified (or default) disk. Make sure you specified the correct file. Then reenter the COPY command, making sure that the disk containing the source file is in the correct drive and that the proper directory (as indicated by a path name, if required) is accessed.

```
Insufficient disk space
     0 File(s) copied
```

EXPLANATION: You entered an invalid destination file name or there is insufficient space on the destination disk for the file you are attempting to copy. If the destination file name you entered was invalid, reenter the COPY command using a valid file name.

If there is not enough disk space to contain the destination file, use another disk or specify an alternate destination disk when you reenter the COPY command.

```
Invalid directory
```

EXPLANATION: At least one of the directories in a specified path does not exist. Check the directory path you specified and reenter the COPY command.

```
Invalid drive name
```

EXPLANATION: The source drive you specified either does not exist or is not configured for your system. Reenter the COPY command using a valid source drive name.

```
Invalid number of parameters
```

EXPLANATION: You entered the COPY command without specifying the minimum required parameters, or you entered the COPY command with too many parameters.

You must always enter a source file specification as part of the COPY command. If you did not specify the minimum required parameters, reenter the command, specifying the required source file and other parameters as needed for the operation you wish to effect.

If you entered too many parameters (if you entered more than a source file specification, a destination, and the optional Verify switch), reenter a valid COPY command line.

```
Not ready error reading drive d
Abort, Retry, Ignore?
```

EXPLANATION: The destination drive you specified either does not contain a disk, the drive door is open, or the drive does not exist or is not configured for your system. If the drive does not exist or is not configured for your system, enter **A** to abort the attempted operation. The system prompt will be displayed. You may then reenter the COPY command using a valid destination drive name.

If the drive does not contain a disk or if the drive door is open, insert a disk and/or close the drive door and then enter **R**. The system will reattempt the copy operation.

CTTY (Resident)

Purpose

The CTTY (Change TTY device) command allows you to change the device from which you issue commands. In most instances, the TTY device is your keyboard and screen.

Entry Form

`CTTY` ***device***

where ***device*** can be any one of the following device names:

AUX — refers to an auxiliary device

CON — refers to console (input from your keyboard, and output to your screen)

COM1 — refers to a device connected to serial port number 1

COM2 — refers to a device connected to serial port number 2

Preliminary Concepts

This command is useful if you want to change the device on which you are working to another. For example, the command

```
CTTY AUX
```

changes all command I/O (input/output) from the current device (the console keyboard) to the AUX port, which is used by such devices as modems and remote terminals. The command

```
CTTY CON
```

moves I/O back to the original device, in this instance, the console. Refer to the section, Files in Chapter 1, "Introductory Concepts," for more information on device names used with the CTTY command.

One condition must be met by any device that you name. You must be sure that any device you name is capable of both input *and* output functions. An example of a device that you could not name, would be a printer, because MS-DOS will try to read from it—and cannot.

Command Line Entry

The CTTY command is entered, followed by the name of the device you wish to transfer command to.

For example, entering

```
CTTY CON
```

and pressing RETURN, will reset the standard input to your keyboard, and output to your screen.

NOTE: When one CTTY command is used to redirect I/O to another device, you must type CTTY CON at the other device to regain control after the session is ended.

Advanced Concepts

The CTTY command will accept the name of any (character-oriented) device, thus allowing you to install your own custom-designed device drivers and to specify the device name.

DATE (Resident)

Purpose

To display and/or change (set) the date known and used by the system.

Entry Forms

```
DATE
DATE mm-dd-yy
```

where ***mm*** is a value from 1 to 12, designating the month;
dd is a value from 1 through 31, designating the day of the month; and
yy is a value from 80 through 99 or 1980 through 1999, designating the year. To designate dates during 2000 through 2099, enter all four numbers in the year—for example, *2083*.

Preliminary Concepts

The date displayed and/or specified with the DATE command is used by the system in "housekeeping" functions in all directory levels. The date is recorded in the directory for any files created or changed, enabling the user to keep track of current files. The DATE command may be invoked from either the terminal or from a batch file in two ways: interactive entry, wherein the system prompts you for the required input; or command line entry, wherein you input the date when you invoke the command.

Using DATE and TIME, the system keeps an accurate internal calendar and clock for as long as the system is running. MS-DOS correctly updates the time and date as values change during any work session. Leap years are also accounted for automatically in the calendar function. (Refer to TIME in this chapter for information on the TIME command.)

NOTE: When you use an AUTOEXEC.BAT file, MS-DOS will not prompt you for the date (and/or time) unless you have included the DATE (and/or TIME) command as part of the AUTOEXEC.BAT file. For information on AUTOEXEC.BAT files, refer to Chapter 5, "Command Features."

Interactive Entry Prompt and Response

If you enter **DATE** with no parameters and press **RETURN**, the system prompts

```
Current date is day mm-dd-yy
Enter new date:
```

In this display, *day* is the day of the week, and *mm-dd-yy* is the date in month, day, year form.

If you do not wish to change the date displayed, press **RETURN**. The system prompt will be displayed. The system will continue to use the displayed date in directory maintenance functions.

If you wish to change the date to a new or current value, enter the desired date and press **RETURN**. The system will store the date entered and use it for all directory maintenance functions until a new date is entered or until the system automatically updates the date.

The allowable parameter values for the date you enter are:

mm = a value from 1 through 12, designating the month;
dd = a value from 1 through 31, designating the day of the month; and
yy = a value from 80 through 99 or 1980 through 1999, designating the year. To designate dates during 2000 through 2099, enter all four numbers in the year—for example, *2083*.

The day of the week displayed by the DATE prompt is calculated automatically by the system, and is not an enterable parameter.

The values entered for the month, day, and year parameters must be separated either by hyphens (-) or slashes (/). If an invalid parameter value or separator is entered, the date entered is ignored and the system prompts

```
Invalid date
Enter new date:
```

You must then reenter a valid date or press **RETURN** to use the preexisting date.

Command Line Entry

If you invoke the DATE command by entering **DATE *mm-dd-yy*** at the system prompt and pressing **RETURN**, the system will store and use the date you enter without displaying the preexisting date or prompting you to enter the date. After the command is executed, the system prompt will be displayed.

The allowable parameter values for the date you enter on the command line are:

mm = a value from 1 through 12, designating the month;
dd = a value from 1 through 31, designating the day of the month; and

yy = a value from 80 through 99 or 1980 through 1999, designating the year. To designate dates during 2000 through 2099, enter all four numbers in the year—for example, *2083*.

The day of the week displayed by MS-DOS as part of the current date is calculated automatically by the system, and is not an enterable parameter.

The values entered for the month, day, and year parameters must be separated either by hyphens (-) or slashes (/).

Usage

The dates appearing in directory displays are a direct result of the DATE command function. If you generate several versions of a file, the date recorded in the directory for each tells you which file is the most current. The dates recorded in the directory are useful for disk maintenance, enabling you to review disk contents and erase files that are no longer needed or that are outdated.

You may wish to use the DATE command simply to change the date that will be recorded for any files that are edited or created during the current work session. MS-DOS will use any date you enter, as long as the parameter values and separators used are valid.

The DATE command may be used with bootable disks that include an AUTOEXEC.BAT file. When this type of disk is booted up and the AUTOEXEC.BAT file executed, the system does not automatically produce the boot prompts for the date and time. If you wish to set the date at bootup, you must include the DATE command in the AUTOEXEC.BAT file. When you create the file, simply write DATE or DATE mm-dd-yy at the appropriate point in the file. (See Chapter 5, "Command Features.")

Error Message

```
Invalid date
Enter new date:
```

EXPLANATION: You have entered an invalid date. You may have entered too much or too little information, an invalid parameter value, or used invalid separators. Reenter the date.

DEL or ERASE (Resident)

Purpose

Deletes (erases) specified file(s) from the default or other specified disk.

Entry Forms

```
DEL filespec
DEL [d:]pathname[\filename]
ERASE filespec
ERASE [d:]pathname[\filename]
```

where ***filespec*** is the file specification of the file (or files) to be deleted,
d is the drive name identifying the disk on which the file to be erased is located,
pathname is the directory path the system must use to locate the desired file or files, and
filename is the primary file name (and extension, if any) of the file to be deleted.

Command Line Entry

You may enter either DEL or ERASE to invoke this command. As a minimum, you must also enter a file name as the *filespec* to be deleted. If you enter only a file name, the system will search the current directory on the default disk for the file you specified, and erase that file.

When you enter a drive name (*d*) as part of the target *filespec*, the system will search the current directory of the specified disk for the file name you enter.

You must enter a *pathname* as part of the file specification if you wish to delete a file from a directory other than the current directory of the default or specified disk. The path name should be followed by a valid *filename*; otherwise *all* files in the specified directory will be deleted.

Deleting a File from the Default Disk

Suppose you have a data file on your default disk that you no longer need. If the file is in the current working directory, you could delete it by entering

```
DEL GARBAGE
```

and pressing **RETURN**. The system will erase the file from the disk and then the system prompt will be displayed.

If the file you wish to delete is not in the current working directory, you must specify the correct path name for the system to locate the file. For example, if the file GARBAGE is in the directory DUMP on the default disk, you could delete it by entering

```
DEL \USER\DUMP\GARBAGE
```

and pressing **RETURN**. In this example, USER and DUMP are assumed to be existing subdirectories. The system will search the specified directory on the default disk for the file and erase it. Then the system prompt will be displayed.

Deleting a File from Another Disk

If you wish to delete a file that is not on the default disk, you must enter the correct drive name as part of the file specification immediately following the DEL command. If the file TRASH.BIN is in the current directory of the disk in drive B, and the disk in drive A is the default, you could delete the file by entering

```
DEL B:TRASH.BIN
```

and pressing **RETURN**. The system will search the current directory of the disk in drive B and erase the file. Then, the system prompt will be displayed.

If the file you wish to delete is not in the current directory of the disk, you must specify the correct path name after the drive name of the disk on which the file is located. For example, if the file TRASH.BIN is in subdirectory LARGE on disk B, you could erase it by entering

```
DEL B:\USER\FIND\LARGE\TRASH.BIN
```

and pressing **RETURN**. In this example, USER, FIND, and LARGE are all existing subdirectories. The system will search the specified directory on the specified disk for the file and erase it. Then, the system prompt will be displayed.

Using Wildcard Characters with the Delete Command

Wildcards may be used as part of the *filespec* with the DEL command. In this way you can delete multiple files in one operation. For example, if you had the files GARBAGE.BIN and TRASH.BIN in the current directory of the default disk and wished to erase them, you could enter

```
DEL *.BIN
```

and press **RETURN**. The system will delete all files with the extension .BIN. In a situation such as this, you should be certain that no files that you wish to keep have the same extension as those you are deleting.

You may similarly delete a group of files that have the same file name and different extensions, or that have a specified character in a certain position, etc. The latter is accomplished by using the ? wildcard character. Refer to Chapter 1 for more information on using wildcard characters in file names.

The system includes a check to help make sure that you do not accidently delete all files from a directory when you use wildcards. If you enter *.* as the file name to be deleted, the system will display the following prompt before executing the command:

```
Are you sure (Y/N)?
```

If you actually wish to delete all files in the current or specified directory, enter **Y** and press **RETURN**. Otherwise, enter N and press RETURN—the delete operation will be aborted and the system prompt will be displayed.

Error Messages

```
File not found
```

EXPLANATION: When this message is displayed, one of the following has occurred:

- a nonexistent file name was entered
- an invalid file name was entered
- the file specified is not in the current directory
- the file specified is not on the disk (default or specified)

Check the parameters you specified and reenter the DEL command.

DEL or ERASE

```
Invalid directory
```

EXPLANATION: At least one of the directories in a specified path does not exist. Check the directory path you specified and reenter the DEL command.

```
Invalid number of parameters
```

EXPLANATION: You invoked the DEL command without specifying at least a file name to be deleted. (That is, you entered simply DEL or ERASE and pressed RETURN.) Reenter the command, specifying the file to be deleted.

```
Not ready error reading drive d
Abort, Retry, Ignore?
```

EXPLANATION: When you entered the DEL command, you entered the drive name for a drive that either does not contain a disk, the drive door is open, or the drive does not exist or is not configured for your system. If the drive does not exist or is not configured for your system, enter **A** to abort the attempted operation. The system prompt will be displayed. You may then reenter the DEL command using a valid destination drive name.

If the drive does not contain a disk or if the drive door is open, insert a disk and/or close the drive door and then enter **R**. The system will reattempt the DEL operation.

DIR (Resident)

Purpose

Displays all or selected entries in the current or specified directory. Unless you use the Wide Display switch, the directory display includes the file size in bytes and the date and time of the last file modification for each file name in the directory.

The DIR command displays one directory at a time. If you wish to display all directory levels, you must use the SEARCH command, as described elsewhere in this chapter.

Entry Forms

```
DIR [d:][filename][/x]
DIR [d:][pathname][/x]
```

where ***d*** is the drive name identifying the disk on which the directory you wish to display is located,
filename is the primary file name and extension, if any, identifying the file or group of files (if wildcard characters are used) for which you wish to display directory information,
pathname identifies the directory you wish to display, if other than the current directory on the default or specified disk, and
/x is one or both of the following switches:
/P, the Page mode switch, or
/W, the Wide Display mode switch.

Command Line Entry

The parameters that may be entered as part of the DIR command line are described below. If none of the parameters are used (in other words, if you enter **DIR** and press **RETURN**), the system will list all entries in the current directory of the default disk.

Drive Name

You must enter a valid drive name (*d*) if you wish to display directory information for a disk other than the default. If you enter no drive name, the default is assumed.

File Name

If you wish to display directory information for a single file or for a selected group of files, enter a *filename* as part of the DIR command. The *filename* may be a primary file name (and extension, if one exists) if you wish to display directory information for a single file. The *filename* may include wildcard characters if you wish to display directory information for a selected group of files.

Path Name

If you wish to display a directory other than the current working directory, you must enter the correct *pathname* for the system to locate the directory. If you do not enter a *filename* as part of the *pathname*, the system will display all directory entries. If you do enter a *filename* as part of the *pathname*, the system will display the directory for the selected file or group of files.

Switches

The two switches described below are supported by the DIR command and may be used either one at a time or both at once. The switches are optional; their use is not required. If both switches are used in one command line, each switch letter must be preceded by a slash (/).

/P—Page Mode

Enter **/P** at the end of the DIR command line to invoke the Page mode directory display. The Page mode enables you to easily review lengthy directories on your screen. When the Page mode is selected, the directory display pauses when the screen is filled. To see the next "page" or screen of the display, press any key.

/W—Wide Display Mode

Enter **/W** at the end of the DIR command line to invoke the Wide Display mode. When the /W switch is entered, file names are displayed in a wide format of five file names per line of the screen display. Only file names are included; other directory information, such as file size and date, is not displayed. This switch is useful when you wish to review the file names in a given directory quickly.

Displaying All Directory Entries

Suppose you are working in the root directory of the default disk. To display the directory, simply enter **DIR** and press **RETURN**. The screen displays all file names in the directory, along with file size in bytes and the date and time of last modification for each file. A typical root directory display is shown in Figure 11.9. Notice that the display includes the names of subdirectories, which are identified by <DIR> in the column that in other cases (that is, for files that are *not* directories) shows file size.

If drive A were the current default and you wished to display the current directory of the disk in drive E, you could do so by entering

```
DIR E:
```

at the system prompt and pressing **RETURN**. The current directory of the disk in drive E would be displayed.

```
 Volume in drive A is DISK1
 Directory of A:\

COMMAND  COM    15464   5-10-83   3:25p
DEBUG    COM    11764   5-11-83   9:10a
EDLIN    COM     4389   5-11-83   8:25a
CHKDSK   COM     6330   5-11-83   8:08a
SYS      COM     2054   5-11-83   9:02a
FORMAT   COM    11546   5-09-83  12:36p
CONFIGUR COM     9155   5-09-83   9:11a
DSKCOPY  COM    13274   5-09-83  12:33p
DSKCOMP  COM     1539   5-09-83   9:16a
MAKE     COM    14490   5-09-83  12:32p
PRINT    COM     5748   5-09-83   8:46a
MASM     EXE    74112   4-01-83   8:03p
LINK     EXE    42368   1-06-83   4:36p
LIB      EXE    32128   1-31-83  12:58p
CREF     EXE    13824   6-02-82   6:06p
EXEFIX   EXE    11776   1-19-83   9:42a
EXE2BIN  EXE     1649   5-11-83   8:34a
RDCPM    COM     3776   5-05-83   8:02a
EXPDIR       <DIR>      6-05-83   9:00a
OMAR         <DIR>      6-11-83   4:45a
      20 File(s)       6144 bytes free
```

Figure 11.9. Typical Root Directory Display

Unless you specify another directory, the DIR command will always display the current working directory of the default or specified disk. To display another directory, you must enter the path name for the system to locate that directory. For example, suppose you are working in the root directory of the default disk and wish to review a user's directory at another level. Suppose the name of the directory you wish to review is PAYROLL and that it is accessed through the root directory and subdirectories USER and ACCOUNTS. To display the directory, you would enter

```
DIR \USER\ACCOUNTS\PAYROLL
```

at the system prompt and press **RETURN**. The screen displays the PAYROLL directory. There is no difference between a subdirectory display and a root directory display with regard to the format and types of information provided in the display.

Displaying Selected Files

There may be times when you want to display directory information for only one file, such as when you need to check file size before copying a file to another disk. Suppose you have a file named BIGFILE.DAT in the current directory of your default disk and wish to display the directory information for that file. To do so, you would enter

```
DIR BIGFILE.DAT
```

at the system prompt and press **RETURN**. The screen displays the directory entry for that file *only*.

You may also display the directory for a selected group of files. Suppose that on your default disk you have stored data files, as well as program files, and that you wish to review the data files, all of which have the extension .DAT. To review the disk's data file content, you could enter

```
DIR *.DAT
```

at the system prompt and press **RETURN**. All files in the current directory that have the extension .DAT will be displayed, regardless of their file names. You may similarly invoke a directory display of files with a given extension and/or use the ? wildcard to display other groups of files. Refer to Chapter 1 for more information on using wildcards in file names.

For your convenience, note the following equivalent uses of the DIR command without and with wildcard characters in the file specification:

COMMAND	EQUIVALENT
DIR	DIR *. *
DIR filename	DIR filename.*
DIR .ext	DIR *.ext

Displaying Subdirectories of the Root Directory

When the current directory of the default or specified disk is a subdirectory, you may use the . and .. notation to display directories. If you are working in a subdirectory of the default disk and enter

```
DIR .
```

and press **RETURN**, the current directory will be displayed in its entirety.

If you are working in a subdirectory of the default disk and enter

```
DIR ..
```

and press **RETURN**, the *parent* directory of the current directory will be displayed in its entirety. For more information on MS-DOS directory structure, refer to Chapter 7, "Directory Features."

Using the Page Mode Switch

Suppose you have a disk on which you have stored many data files. If there are too many files to fit on one screen display, you may select the Page mode so that the directory display will pause when the screen is filled. To do so, simply use the /P switch at the end of the DIR command line, as in the example below.

```
DIR /P
```

This command will display the first screen of the current directory for the default disk, then prompt you to press any key to see the next screen of the directory.

You may use the /P switch as part of any DIR command line. The switch may be used at the same time as the /W (Wide Display) switch.

Using the Wide Display Mode Switch

The /W switch may be used to "condense" directory displays. When you enter /W at the end of a DIR command line, you invoke the Wide Display mode. In this mode, only file names are displayed, with five file names per line of the screen display. Other directory information, such as the size of files and the date of their last modification, is not included. This switch is useful when you wish to review file names in a given directory quickly.

The /W switch may be used as part of any DIR command line and may be used in conjunction with the /P (Page mode) switch. A typical wide directory display is shown in Figure 11.10.

```
 Volume in drive A is DISK1
 Directory of A:\

COMMAND  COM   DEBUG    COM   EDLIN    COM   CHKDSK   COM   SYS     COM
FORMAT   COM   CONFIGUR COM   DSKCOPY  COM   DSKCOMP  COM   MAKE    COM
PRINT    COM   MASM     EXE   LINK     EXE   LIB      EXE   CREF    EXE
EXEFIX   EXE   EXE2BIN  EXE   RDCPM    COM
       18 File(s)     6144 bytes free
```

Figure 11.10. Wide Directory Display

Printing the Directory Display

The DIR command gives you an immediate reference to the root directory entries on a disk or to entries in any subdirectory. If you have a printer connected to your microcomputer, you may easily obtain a hard (printed) copy of the directory display by using input/output redirection. For example, to obtain a printed copy of the current directory of the default disk, you could enter

```
DIR >PRN
```

and press **RETURN**. The system would send the output of the DIR command (that is, the directory that otherwise would be displayed on the screen) to the printer. Printing directory information provides you an easily-referenced index to disk contents.

Error Messages

```
File not found
```

EXPLANATION: When you invoked the DIR command for a directory other than the current directory on the specified disk or for a given file, the system could not locate the directory or file that you specified. Reenter the DIR command, and be sure that you correctly specify the directory and/or file.

```
Invalid directory
```

EXPLANATION: The path name you specified is not a valid directory path name or does not exist. Reenter the DIR command, making sure that you enter a valid path name.

DISKCOMP (Transient)

Purpose

Makes a track-by-track comparison of two disks to find out if the contents of the two disks are identical or different.

Entry Form

```
DISKCOMP[ s: [d:]]
```

where *s* is the Source1 drive name; and
d is the Source2 drive name.

Preliminary Concepts

DISKCOMP compares data on two disks to determine if the data is identical on both disks. DISKCOMP and DISKCOPY can only be used on disks that are both the same format. Also, both disks must have the same number of sides, tracks per inch and density. The DISKCOMP utility can be used to verify that the DISKCOPY utility makes an exact duplicate of a source.

The same type of comparison is run when you use the /V switch with DISKCOPY. The /V switch is used to verify that the data is copied correctly during the DISKCOPY procedure.

Interactive Entry Prompts and Responses

There are two methods available for invoking DISKCOMP. One method is to enter the command (DISKCOMP) and sources on the command line. The other method is to enter only the DISKCOMP command, which then prompts you for the sources. For example, if you enter

```
DISKCOMP
```

and press **RETURN**, the screen displays

```
            DISKCOMP version 2.xx


Source1 drive name?  (A-x) _:
```

After responding, you will be asked:

```
Source2 drivename?  (A-x) _:
```

NOTE: *x* is a variable for the drive depending on your system configuration. If you need to know more about how your drives are designated refer to Chapter 6, "Bootup Results."

CAUTION: Source1 and Source2 must be designated as *logically* different disk drives. The disks must be the same size and format and have the same number of usable sides. (Refer to Chapter 5, "Command Features," for details regarding single drive systems.)

NOTE: The MS-DOS version numbers are shown in the screen displays as the variable, 2.xx. This is because your screen display may be a more recent version release than that available when this manual was prepared.

After you have entered the Source2 drive name, DISKCOMP will prompt you:

```
Place Source1 disk in drive s and Source2 disk in d

Press RETURN when ready.
```

where *s* is the letter of the Source1 drive; and
d is the letter of the Source2 drive that you specified in the first two prompts.

After you have placed the correct disks in the specified drives and pressed RETURN, DISKCOMP will tell you that it is working with the message

```
Verifying...
```

After a few moments (when the comparison has been successfully completed) you should see the message:

```
Do you wish to compare more disks (Y/N)? <N>
```

This indicates that the disks compared are identical. Press **RETURN** or **N** and DISKCOMP will return to the system prompt. Press **Y** (and press **RETURN**) if you wish to compare additional disks. If you get a different message, it means that there was some verification error and the disks are not identical (refer to Error Messages).

Command Line Entry

The command line entry method of invoking DISKCOMP requires that you enter command line parameters for the Source1 and Source2 drives. If you only enter one drive letter on the command line, that drive is assumed to be Source1, and you will be prompted for the Source2 drive specification. For example, if you enter

```
DISKCOMP A:
```

and press **RETURN**. The screen will display:

```
                DISKCOMP version 2.xx


Source2 drive name?  (A-x)  _:
```

After entering Source2, DISKCOMP will prompt you:

```
Place Source1 disk in drive A and Source2 disk in d
Press RETURN when ready.
```

NOTE: In this message the *d* reflects the drive name that you entered as Source2.

If you enter

```
DISKCOMP A: B:
```

and press **RETURN**. DISKCOMP displays:

```
                DISKCOMP version 2.xx


Place the Source1 disk in A and the Source2 disk in B
Press RETURN when ready.
```

Advanced Concepts

The DISKCOMP command is useful if you have used DISKCOPY to create a backup disk and wish to double check to make certain that the copy is correct. If you store a master and a backup copy of data disks and the backup was created with DISKCOPY, you can use DISKCOMP to verify that the disks are still identical and that their data content has not degraded after weeks, months, or years. The key to this, of course, is that the backup was created from the master with DISKCOPY. Otherwise, individual disk files may need to be verified with FC (the File Compare utility).

Error Messages

```
Disk verify failure
```

EXPLANATION: The disks you were comparing are *not* identical.

```
Drives specified must not be the same
```

EXPLANATION: You specified the same drive name for Source2 as you did for Source1. DISKCOMP will redisplay the prompts, beginning with:

```
Source1 drive name? (A-x) _:
```

You must reenter the drive names, being careful that Source1 and Source2 are not the same drive.

```
Incompatible media. Cannot continue
```

EXPLANATION: This error message will appear if you try to run DISKCOMP on two disks that are of a different type. (For example, comparing a double-sided disk with a single-sided disk will generate this error, even though the data on the disks is the same.) You cannot use DISKCOMP to compare incompatible media.

```
Read error on Source1 drive
```

EXPLANATION: The disk in this drive is either a non-DOS disk or has bad sectors.

```
Read error on Source2 drive
```

EXPLANATION: The disk in this drive is either a non-DOS disk or has bad sectors.

DISKCOPY (Transient)

Purpose

Copies the contents of one disk onto another disk so that the contents of both are identical.

Entry Form

```
DISKCOPY[ s: [d:]][/V]
```

where **/V** is the switch which sets the verify function on;
s is the source drive name; and
d is the destination drive name.

Preliminary Concepts

The DISKCOPY command can be invoked in two different ways. You may enter all parameters to the DISKCOPY command, which will begin execution as soon as you press RETURN, or you can simply enter DISKCOPY and be prompted for all information.

DISKCOPY will format the destination disk first; once the formatting is completed, DISKCOPY will begin copying the data. After copying, if you specified the /V switch, DISKCOPY will verify that the copy is identical to the source disk. If an error occurs at any time, a message will be displayed and DISKCOPY will terminate. The disk types must be the same in order for DISKCOPY to function. For example, you cannot use DISKCOPY to copy a double-sided disk to a single-sided disk.

Command Line Entry

The first parameter you can specify is the source drive (*s*). The second parameter is the destination drive (*d*). The third parameter is the /V switch. Specify this switch if you want DISKCOPY to verify the copies. If you do not specify a source drive and a destination drive, DISKCOPY will prompt you to enter them.

Examples

The DISKCOPY program may be invoked in two ways. The command line can include the source and destination drives (the Command Line Entry Method), or the source and destination can be omitted (the Interactive Entry Prompt Method.) In the latter case, DISKCOPY prompts you for the source and destination. If you enter

```
DISKCOPY
```

at the system prompt and press **RETURN**. DISKCOPY will display the following:

```
                    DISKCOPY version 2.xx


Source drive name? (A-x) _:
```

Enter the drive name identifying the drive that contains the disk you wish to copy. DISKCOPY continues with:

```
Destination drive name? (A-x) _:
```

Enter the drive name identifying the drive that contains a blank disk (any information on the destination disk will be destroyed by DISKCOPY when it formats the disk).

NOTE: *x* is a variable for the drive depending on your system configuration. If you need to know more about how your drives are designated refer to Chapter 6, "Bootup Results."

DISKCOPY

NOTE: The MS-DOS version numbers are shown in the screen displays as the variable, 2.xx. This is because your screen display may be a more recent version release than that available when this manual was prepared.

DISKCOPY next tells you to place your source disk in the drive you specified and to place the destination disk in its specified drive:

```
Place the source disk in s and the destination disk in d

Press RETURN when ready.
```

Press **RETURN** when you have placed the correct disks in the drives specified. DISKCOPY tells you when it begins each of its operations. For example, you will see

```
Formatting destination...
```

and

```
Copying...
```

appear on your screen. When all operations have finished, DISKCOPY asks you

```
Do you wish to copy another disk (Y/N)? <N>
```

Pressing any key except Y (followed by a **RETURN**) ends DISKCOPY and returns you to the system prompt. Entering **Y** and pressing **RETURN** causes DISKCOPY to request source and destination again.

NOTE: Refer to Chapter 5, "Command Features," for details regarding single drive systems.

If the source and destination drives are specified in the command line, as in the Command Line Entry Method, the entry components will be as follows. Enter

```
DISKCOPY A: B:
```

at the system prompt and press **RETURN**. DISKCOPY does not prompt you for source and destination but begins execution and displays:

```
                DISKCOPY version 2.xx


Place the source disk in A and destination disk in B

Press RETURN when ready.
```

When you have placed the correct disks in the correct drives and have pressed **RETURN**, DISKCOPY informs you of its current operation:

```
Formatting destination...
```

and

```
Copying...
```

When DISKCOPY has completed its operations, it will return to the system prompt. (DISKCOPY only asks `Do you wish to copy another disk?` when you do not specify source and destination on the command line.)

If you specify only a single drive name on the command line, such as **DISKCOPY C:**, DISKCOPY assumes that the specified drive is the source and then prompts you for the destination.

The /V switch is available as an option. The /V switch must be entered directly after DISKCOPY or the drive names (if entered). This switch causes DISKCOPY to verify that the disk has been copied correctly. When the /V switch has been selected, DISKCOPY will verify the copied data after it formats and copies. The corresponding message is:

```
Verifying...
```

If you do not use the /V switch, you may use the DISKCOMP command after DISKCOPY if you wish to verify that the copy is identical to the original.

NOTE: Because DISKCOPY creates an exact duplicate of the original, the data in each physical disk sector is identical. Since the formatting operation that occurs under DISKCOPY does not *gather* the bad sectors of the destination disk, so they cannot be used. This means that DISKCOPY cannot copy properly if the destination disk has a bad sector. If a bad sector error does occur, it is wise to reformat the disk with the FORMAT program (using the /V switch). That disk should then only be used with programs other than DISKCOPY.

CAUTION: Make certain you have a backup copy or an original copy of any disk that you plan to use as a destination disk for DISKCOPY, and make sure that you place the destination disk in the destination drive and the source in the source drive. If you inadvertently switch the two, your source disk will be erased.

Error Messages

```
Disk copy failure
```

EXPLANATION: The destination disk could not be copied to. This could mean bad sectors or other disk media problems exist.

```
Disk verify failure
```

EXPLANATION: The destination disk did not take the copy correctly. This could mean bad sectors exist.

```
Drives specified must not be the same
Source drive name? (A-x) _:
```

EXPLANATION: This message will appear if you try to specify the same drive name for the source *and* the destination. You must reenter valid drive names.

```
Format failure on destination drive
```

EXPLANATION: The destination disk is bad and cannot be formatted.

```
Incompatible media.
Cannot continue.
```

EXPLANATION: You tried to run DISKCOPY from a source disk that is different from the destination. For example, you would get this error if you tried to use DISKCOPY to copy a floppy disk to a Winchester partition.

```
Read error on destination drive
```

EXPLANATION: During the verification process the destination disk could not be read. The destination disk should be copied again; if the error persists, the destination disk may have bad sectors.

```
Read error on source drive
```

EXPLANATION: The disk in the source drive may have bad sectors or other media problems.

```
Write error on destination drive
```

EXPLANATION: The destination disk may be write-protected or may have bad sectors.

ECHO (Resident, Batch-Processing)

Purpose

The ECHO command is used to control the display of commands and remarks (or messages) during the execution of a batch file.

Entry Forms

```
ECHO [ON]
ECHO [OFF]
ECHO [message]
```

where **ON** turns on the ECHO command;
OFF turns off the ECHO command; and
message is any comment or remark you want displayed as the batch file executes. This feature of ECHO is similar to the REM command (remark), except that ECHO displays only the *message*, while the REM command displays 'REM' followed by the *message*.

Preliminary Concepts

The resident, batch-processing commands are most often executed from within a batch file, although they may be used directly from the command line in some instances. Refer to Chapter 5, "Command Features," for a complete explanation of batch file creation and execution.

NOTE: You must always end each line within a batch file by pressing the RETURN key.

The ECHO command may be used in one of three ways:

- To turn the display of the batch file commands on or off.
- To display comments (*message*) during the execution of a batch file.
- To display the status of the ECHO command (whether it is off or on).

Normally, commands in a batch file are displayed or echoed on the screen as they are executed by the system. ECHO OFF stops the display of commands and remarks. When ECHO is set off, nothing in the batch file will be displayed except error messages and any *message* specified with the ECHO command. ECHO ON, turns the command display on.

If ECHO is entered without any parameters (that is, if ON, OFF, or *message* are not specified), the current setting of ECHO is displayed.

NOTE: The default setting of the ECHO command at bootup is ON.

Command Line Entry

The ECHO command is entered followed by either ON or OFF or any *message*, up to a total line length of 127 characters.

Any *message* that you enter with this command will be displayed on the screen, regardless of whether ECHO is *ON* or *OFF*.

NOTE: Using the ECHO command does not interfere with error messages that are displayed during command execution.

Turning Off the Command Display

In the following example, the ECHO command is used to stop the display of the command line entry while still displaying the information desired. For this example the batch file contains the following commands:

```
ECHO off
DIR A:
```

When the batch file containing these commands is executed, the screen will display the following:

```
ECHO off

Volume in drive A has no label

Directory of A:\

filenam1.ext
filenam2.ext
filenam3.ext

3 file(s) n bytes free
```

In the example, ECHO off is displayed, but the line containing the command DIR A: is not displayed because ECHO is set off. The output of the DIR command, however, is unaffected.

If you had set the ECHO command on by entering

```
ECHO ON
DIR A:
```

and pressing the **RETURN** key, the DIR A: command would have been displayed.

Displaying Messages During Batch File Execution

You may also use the ECHO command to enter a message you want displayed during batch file execution. For example, if you want the message, "This is a batch file," displayed during the execution of a batch file, you would include the line

```
ECHO This is a batch file
```

within the batch file. When the batch file is executed, the message

```
This is a batch file
```

will be displayed.

Displaying the Status of the ECHO Command

To have your screen display the current status of the ECHO command, simply type

```
ECHO
```

and press the **RETURN** key. The screen will then display

```
ECHO is on
```

if the ECHO command is set on, or

```
ECHO is off
```

if the ECHO command is set off.

EXE2BIN (Transient)

Purpose

Converts .EXE (executable) files to binary format. This results in a saving of disk space and faster program loading.

Entry Form

```
EXE2BIN filespec [d:][filename[.ext]]
```

where ***filespec*** is the name of the .EXE file you wish to convert to a binary image;
d is the drive name identifying the drive to which you want the binary output sent;
filename is the optional name of the output binary image;
and ***.ext*** is the optional extention of the output binary image.

Preliminary Concepts

This command is useful only if you want to convert .EXE files to binary format. The file named by *filespec* is the input file. If no extension is specified, it defaults to .EXE. The input file is converted to .COM file format (memory image of the program) and placed in the output file. If you do not specify a drive, the drive of the input file will be used. If you do not specify an output filename, the input filename will be used. If you do not specify a filename extension in the output filename, the new file will be given an extension of .BIN.

The input file must be in valid .EXE format produced by the linker. The resident, or actual code and data part of the file must be less than 64K. There must be no STACK segment.

Two kinds of conversions are possible, depending on whether the initial CS:IP (Code Segment:Instruction Pointer) is specified in the .EXE file:

1. If CS:IP is not specified in the .EXE file, a pure binary conversion is assumed. If segment fixups are necessary (i.e., the program contains instructions requiring segment relocation), you will be prompted for the fixup value. This value is the absolute segment at which the program is to be loaded. The resulting program will be usable only when loaded at the absolute memory address specified by a user application. The command processor will not be capable of properly loading the program.

2. If CS:IP is specified as 0000:100H, it is assumed that the file is to be run as a .COM file with the location pointer set at 100H by the assembler statement ORG; the first 100H bytes of the file are deleted. No segment fixups are allowed, as .COM files must be segment relocatable. Once the conversion is complete, you may rename the resulting file with a .COM extension. Then the command processor will be able to load and execute the program in the same way as the .COM programs supplied on your MS-DOS disks.

Error Messages

```
File cannot be converted
```

EXPLANATION: This message is displayed if the source file you specified is not in the correct format. That is, the executable file you specified cannot be converted to binary format because CS:IP does not meet either of the criteria described under Preliminary Concepts, or because it meets criterion 2 but has segment fixups. This message is also displayed if the file you specified is not a valid executable file.

```
File not found
```

EXPLANATION: The input file is not on the disk specified.

```
Insufficient memory
```

EXPLANATION: There is not enough memory to run EXE2BIN.

```
File creation error
```

EXPLANATION: EXE2BIN cannot create the output file. Run CHKDSK to determine if the directory is full, or if some other condition caused the error.

```
Insufficient disk space
```

EXPLANATION: There is not enough disk space to create a new file.

```
Fixups needed - base segment (hex):
```

EXPLANATION: The source (.EXE) file contained information indicating that a load segment is required for the file. Specify the absolute segment address at which the finished module is to be located.

```
WARNING -Read error on EXE file.
Amount read less than size in header
```

EXPLANATION: This is a warning message only. It suggests that the file on disk is smaller than the size indicated by the header in the EXE file. This could mean a bad .EXE file was input. Produce the .EXE file again and rerun EXE2BIN.

EXIT (Resident)

Purpose

Allows you to exit COMMAND.COM (the command processor). Control is then returned to a previous level (such as an application program), if one exists.

Entry Form

```
EXIT
```

Preliminary Concepts

This command can be used when you are running an application program and want to start the MS-DOS command processor, and then return to your program. For example, you may wish to examine a directory on the disk in drive B while running an application program. To accomplish this, you must request your application program to start the command processor. This will cause your system prompt to display. (Refer to the explanations of COMMAND.COM in Chapter 9, "System Component Features," for more information on starting the command processor.)

After your system prompt displays, you can type the DIR command, and MS-DOS will display the directory for the disk in the default drive. (You may also execute any other MS-DOS command, since the operating system command processor is in control at this point.) To return to the previous level (your application program), type EXIT.

Command Line Entry

The EXIT command is invoked by entering

```
EXIT
```

and pressing **RETURN**.

FC (Transient)

Purpose

Used to compare one file to another to see if they match and to create a file that contains a list of differences (if any) between the two files.

Entry Form

```
FC[/x] filespc1 filespc2
```

where ***filespc1*** is the file specification of the first file of the two you wish to compare,

filespc2 is the file specification of the second file of the two you wish to compare, and

/***x*** is one or more of the following optional switches:

/B	Binary—Forces FC to make a binary, byte-by-byte comparison of the specified files. No attempt is made to resynchronize after (if) a mismatch is found.
/C	Case ignored—Causes the case of letters within the specified files to be ignored during comparison. All letters in the files are considered to be uppercase.
/#	Number of lines for match—Specifies the number (from 1 to 9) of lines required to match for the compared files to be considered as matching again after a difference has been found.
/W	Whites compressed—Causes FC to compress white space (tabs and spaces) during comparison of the files, such that multiple contiguous spaces or tabs in any given line are considered as a single space.

Preliminary Concepts

It is sometimes useful to compare files on your disk(s). If you have copied a file and later want to compare the original file and the copy to see which is the most current, you can use the FC (file comparison) utility.

FC compares the contents of two files that you specify. The files may be on the same or different disks, and/or in the same or different directories. In essence, FC compares the second file you specify to the first file you specify. If any differences between the two files are found, FC points to the place in the first file where information begins to differ and lists the differences in the files. The differences between the two files can be output to the display screen or to a file. You can then check to see what the differences are so that you can decide which file you wish to save..

You can use FC to compare either source files (files containing source statements of a programming language) or binary files (files output by an assembler, linker, or high-level language compiler). The comparison may be made in one of two ways: on a line-by-line basis or on a byte-by-byte basis.

When performing a line-by-line comparison, FC isolates the blocks of lines within the files that are different and displays those blocks of lines. When performing a byte-by-byte comparison, FC isolates the individual bytes within the files that are different and displays those bytes. FC normally performs line-by-line comparisons for source files; and byte-by-byte comparisons for binary files. However, you can force a binary comparison of any file by means of an optional switch. Other operational characteristics may also be selected by optional switches. Supported switches are described under Command Line Entry in this section.

During normal execution of a line-by-line comparison, FC reports the differences between the two files you specify by displaying the first file name followed by the lines (if any) that differ between the files, followed by the first line to match in the files after a difference was found. FC then displays similar information for the second file. That is, FC displays the second file name followed by the lines (if any) that differ between the files, followed by the first line to match in the files after a difference was found. The default value for the number of lines that must match (after a difference has been found) in order for FC to consider the files as matching is 3. This value can be changed with the /# switch.

An example showing the form of the difference report is shown below:

```
         ...
         ...

----------<filespc1>
<difference>
<1st line to match file2 in file1>

----------<filespc2>
<difference>
<1st line to match file1 in file2>
------------------------------------

         ...
         ...
```

Throughout the file comparison operation, FC will list each difference found in the form illustrated above. If there are too many differences (that is, if differences found involve too many lines), FC will simply report that the files are different and then terminate. That is, FC displays

```
***Files are Different***
```

and then the system prompt is displayed again.

During execution of a comparison of source files, FC uses all available memory as buffer space to hold the source files. If the source files are larger than available memory, FC only compares as much of the files as it is able to load into memory. If no matches are found in those portions of the files in memory, FC reports that the files are different and then terminates. Whether or not FC determines that the files match, no more of the source files' contents are loaded into memory for comparison.

If binary files are being compared and the files are larger than available memory, FC also uses available memory as buffer space to contain the files being compared. Unlike source files, however, additional portions of the binary files will be read into memory in sequence as needed by FC until the entire files have been compared. All differences are output to the screen (or to a file, if you redirect output) the same as they would be for comparisons of binary files that *do* fit in available memory.

When execution of the comparison is complete, FC exits and the system prompt is displayed.

Command Line Entry

FC command line parameters are described in the paragraphs that follow. As a minimum, you must always enter the command name (FC) and two file specifications for the files you wish to compare. Optionally, you may specify one or more of the supported switches to determine some of the operational characteristics of FC.

File Specifications

The files you specify for comparison may be on the same or different disks and/or in the same or different directories. For each file you wish to compare, you must enter the file specification (*filespc1* and *filespc2*) for FC to locate the file. If a file is not on the default disk, you must include the drive name identifying the disk on which the file is located. If a file is not in the current directory of the default or specified disk, you must include the path name for the directory in which the file is located. The file names you specify must be specific; wildcard characters cannot be used in the file specifications.

The *filespc1* and *filespc2* parameters must be separated from each other and from other command line parameters by a space or equivalent MS-DOS delimiter. Anything entered on the command line following *filespc2* will be ignored.

As an example, suppose you entered

```
FC B:\FOO\BAR\FILE1.TXT \BAR\FILE2.TXT
```

at the system prompt and pressed **RETURN**. FC would locate file FILE1.TXT in the \FOO\BAR directory of the disk in drive B and compare it with file FILE2.TXT in the \BAR directory of the default disk. (Since no drive name is specified for the second file specification, FC searches the default disk for that directory and file.)

Switches

One or more optional switches may be entered as part of an FC command line. The switch(es) must be entered immediately following the command name, before the *filespc1* and *filespc2* parameters are entered. If any switches are entered after the file specifications, they will be ignored. No delimiter other than the MS-DOS switch character (/) normally entered as a part of each switch is required between any switches you enter and the command name.

Switches are provided so that you can force a binary comparison of the files, cause FC to ignore letter case when comparing files, cause FC to compress white space within lines, or specify the number of lines that must match (after a difference has been found) in order for FC to consider that the files match.

Defaults for FC operation when no switches are used are as follows:

- Perform line-by-line comparison for source files and byte-by-byte comparison for binary files.
- Do not ignore letter case when comparing source files.
- After a difference is found in source files, a minimum of 3 lines must match in order for the files to match.
- Do not compress white space within lines when comparing source files.

/B—Binary

Use the /B switch to force a binary comparison of the specified files. In a binary comparison, the files are compared byte by byte, and no attempt is made to resynchronize the files if a difference is found. The difference report produced in a binary comparison takes the following form:

```
--ADDRS----filespc1----filespc2
xxxxxxxx   yy          zz
```

where *xxxxxxxx* is the relative address of the pair of bytes within the files (addresses start at 00000000 at the beginning of the files);
yy is the mismatched byte from *filespec1* and
zz is the mismatched byte from *filespc2.*

During a binary comparison, if one of the files contains less data than the other, a message is displayed. For example, if *filespc1* ends before (contains less data than) *filespc2*, FC displays

```
***Data left in filespc2***
```

/C—Case Ignored

Use this switch when comparing source files to cause FC to ignore the case of letters, such that a given uppercase letter will match the same letter in lowercase. Essentially, this switch causes all letters in the files to be read as uppercase.

For example, when the /C switch is used, the line

```
Much_MORE_data_IS_NOT_FOUND
```

will match the line

```
much_more_data_is_not_found
```

It is often useful to use the /C and /W (Whites compressed) switches together. For example, it you specified both switches when invoking FC, the line

```
_ _ _DATA_was_found_ _ _
```

will match the line

```
data_was_found
```

/#—Number of Lines Required for Match

Use this switch when comparing source files to specify the number of lines that must match after a difference is found in order for FC to consider that the files as a whole match. If you do not use this switch, the default value of three lines is used. The variable # may be specified as any value from 1 to 9.

/W—Whites Compressed

Use this switch when comparing source files to cause FC to compress "whites" (multiple spaces or tabs) within lines such that multiple contiguous spaces or tabs are considered a single space. Note that although FC will compress whites when this switch is used, it will *not* ignore them unless they are at the beginning and/or end of a line. For example, if you used the /W switch, the line

```
_ _ _More_data_to_be_found_ _ _ _
```

will match the line

```
More_data_to_be_found
```

and the line

```
_ _ _ _ _ _ _More_ _ _ _ _ _ _data_to_be_ _ _ _ _found_ _ _ _ _ _ _
```

However, the above three lines will *not* match the following line:

```
_ _ _ _Moredata_to_be_found
```

Example 1

Assume these two ASCII files are on the default disk:

ALPHA.ASM	BETA.ASM
FILE A	FILE B
A	A
B	B
C	C
D	G
E	H
F	I
G	J
H	1
I	2
M	P
N	Q
O	R
P	S
Q	T
R	U
S	V
T	4
U	5
V	W
W	X
X	Y
Y	Z
Z	

To compare the two files and display the differences on the terminal screen, enter

```
FC ALPHA.ASM BETA.ASM
```

at the system prompt and press **RETURN**. FC compares ALPHA.ASM with BETA.ASM and displays the differences on the terminal screen. All other defaults remain intact. (The defaults are: do not use tabs, spaces, or comments for matches; and do a source comparison on the two files.)

The output from the comparison will appear as follows (the Notes do not appear):

```
-----------ALPHA.ASM
D
E
F
----------BETA.ASM
G

---------------------------------
```

NOTE: ALPHA file contains DEFG, BETA contains G.

```
----------ALPHA.ASM
M
N
O
P

----------BETA.ASM
J
1
2
P

--------------------------------
```

NOTE: ALPHA file contains MNO where BETA contains J12.

```
----------ALPHA.ASM
W

----------BETA.ASM
4
5
W
```

NOTE: ALPHA file contains W where BETA contains 45W.

Example 2

Using the same two source files as described in Example 1, you can print the differences on the printer. In this example, suppose you wished to specify that four successive lines must be the same to constitute a match. You could enter

```
FC/4 ALPHA.ASM BETA.ASM >PRN
```

at the system prompt and press **RETURN**. The following would be printed:

```
----------ALPHA.ASM
D
E
F
G
H
I
M
N
O
P

----------BETA.ASM
G
H
I
J
1
2
P
--------------------------------

----------ALPHA.ASM
W

----------BETA.ASM
4
5
W
```

NOTE: P is the 1st of a string of 4 matches.

NOTE: W is the 1st of a string of 4 matches.

NOTE: The use of the > symbol in the previous example, to direct output to the printer (PRN), is an instance of I/O redirection. This procedure is discussed in Advanced Concepts in this section.

Example 3

Using the same two source files used in the previous example, supposed you wished to force a binary comparison and display the differences on your screen. You would enter

```
FC/B ALPHA.ASM BETA.ASM
```

at the system prompt and press **RETURN**. The /B switch forces a binary comparison. This switch and any others must be typed *before* the file names in the FC command line. The following display would appear:

```
--ADDRS----F1---F2--
00000009   44   47
0000000C   45   48
0000000F   46   49
00000012   47   4A
00000015   48   31
00000018   49   32
0000001B   4D   50
0000001E   4E   51
00000021   4F   52
00000024   50   53
00000027   51   54
0000002A   52   55
0000002D   53   56
00000030   54   34
00000033   55   35
00000036   56   57
00000039   57   58
0000003C   58   59
0000003F   59   5A
00000042   5A   1A
```

Advanced Concepts

One of the most useful features of the FC command is the ability to redirect FC output to a file. Ordinarily, the differences and matches between the two files you specify will be displayed on your screen. However, you can redirect the output to a file by using MS-DOS I/O redirection. (Refer to Chapter 8, "Input/Output Features".)

To compare *File1* and *File2* and then send the FC output to a file named DIFFER.TXT in the current directory of the default disk, you could enter

```
FC File1 File2 >DIFFER.TXT
```

at the system prompt and press **RETURN**. The differences and matches between *File1* and *File2* will be put into DIFFER.TXT on the default drive.

Error Messages

`Bad file, Read error in:filespec`

EXPLANATION: FC could not read the entire file. The file may be bad. You should check the file with CHKDSK.

`File not found:filespec`

EXPLANATION: FC could not find the file you specified. The file is probably not on the disk in the designated drive or is not in the specified directory. Check the file specification you entered and make sure that the appropriate disk is available, then reenter the command.

`Incorrect DOS version`

EXPLANATION: You are running FC under a version of MS-DOS that is not version 2 or above. You must use a compatible version of FC.

`Invalid number of parameters`

EXPLANATION: You have specified the wrong number of parameters on the FC command line. Reenter, using the proper number of parameters.

`Invalid parameter:x`

EXPLANATION: One of the switches (shown in the error message in place of the *x*) you specified is invalid. Reenter the command, using a valid switch.

FIND (Transient)

Purpose

Searches for a specific string in one or more specified files. If no file is specified, searches the standard input for the specified string.

Entry Form

```
FIND [/x] "string" [filespec...]
```

where *"**string**"* is the string for which you want FIND to search;
filespec is the file specification (or series of file specifications) of the file(s) to be searched; and
/x is one or more of the following switches:

/C	Count lines (causes FIND to display only the total number of lines that contained a match in each specified file).
/I	Ignore letter case (causes FIND to ignore letter case in executing the search function).
/N	Number lines (causes FIND to precede each displayed line with its relative line number within the file).
/V	Variant lines (causes FIND to display all lines from the specified file(s) that do *not* contain the specified string).

Preliminary Concepts

The FIND command is a filter that accepts as parameters a string and a series of file specifications; it searches the specified files for lines containing a match with the specified string and displays them on the screen. If no files are specified in the FIND command line, FIND searches the standard input and displays all lines that contain the specified string.

This command may easily be used with pipes to afford you greater flexibility in its application. For general information on pipes and filters, refer to Chapter 8, "Input/Output Features."

Command Line Entry

The parameters of the FIND command line are described below. The command line must always include the command name, FIND. Other entry requirements are noted where appropriate in the parameter descriptions.

Search String

When invoking FIND, you must specify the *string* for which you wish the program to search. The string may be any alphanumeric series of characters, and may include spaces. The string may be of any length, as long as the command line as a whole does not exceed the input buffer maximum of 127 characters in a single command line. If optional switches are used, the string must follow the switches and be preceded by a space. If no optional switches are used, the string must follow the command name (FIND) and be preceded by a space.

The string *must* be enclosed in quotation marks (double quotation marks if the string already contains quotation marks) and must be entered exactly as it appears in the file or files you want the program to search. Thus, you must be careful about upper- and lowercase letters, punctuation, and spacing within the string. The string must match a series of characters in the specified file(s) *exactly* in order for the program to recognize it as a match. With regard to letter case, an exception to this requirement can be made by using the /I (Ignore letter case) switch. (Refer to Switches in this section.) If you use the /I switch in a command line, FIND will not require that letter case within the string and within the file(s) be the same; however, in all other respects the string must match a series of characters in the file(s) exactly for FIND to recognize a match.

File Specification

Normally, you will specify one or more files to be searched for the specified string. The *filespec* for the file(s) follows the string and is separated from it by a space. If no file is specified, FIND will analyze input received from the standard input and display all lines that contain the specified string.

A specific *filespec* (or series of file specifications) must be provided; wildcard characters may not be used. If the file you want searched is on the disk in the default drive, you may simply enter the primary file name and extension. If the file is on another disk, you must enter a full file specification, including the appropriate drive name. If the file is not in the current directory of the default or specified disk, you must include the directory path name for the file.

If you wish to specify a series of files to be searched, enter the exact file specification for each file and separate each file specification from the others with spaces. Any number of file names may be specified in the series as long as the total number of characters in the command line does not exceed 127 characters. When you specify a series of files, the files may all be on the same disk or they may be on different disks.

Switches

The use of any of the switches supported by the FIND command is optional. They are provided to allow you to use the command to meet your particular needs at a given time. When switches are used, they must be entered immediately following the command name and before the search string. If they are entered at the end of the command line, the system will not recognize them as switches. Switches should be separated from other parameters on the command line by spaces.

More than one switch may be used in a single command line. However, the /C (Count lines) switch cannot be used with the /N (Number lines) switch. If both switches are used in the same command line then the /N switch is ignored.

When more than one switch is used in a single command line, the only delimiter required between switches is the switch character (/) that is normally entered as part of each switch; no spaces should be used between switches.

/C—Count Lines

Normally, FIND searches the file(s) you specify for the string you specify, and displays on the screen the lines containing a match. When you use the /C switch, FIND displays only the total number of lines in each file containing a match. This display is in the form

```
----------filespec: nn
```

where *filespec* is the file specification and *nn* is the number of matches found. This line is displayed for every file specified in the command line.

/I—Ignore Letter Case

Use this switch to cause FIND to ignore the upper/lowercase distinction for letters in the search string. This switch overrides the default requirement that letters in the search string be entered *exactly* as they may appear in the file(s) to be searched. When the /I switch is used, upper- and lowercase letters are considered equivalents. That is, A is the same as (matches) a, B is the same as b, and so on. When the /I switch is *not* used, FIND will not consider A as a match for a, and so on.

/N—Number Lines

Use this switch to cause FIND to precede displayed lines with their relative line numbers, as applicable to their positions within the file in which they are located. The line number for each line is displayed in square brackets ([ ]). This is useful when you wish to know where a given string occurs within a file so that you can open the file and edit the lines containing the string. For example, with some text editor programs such as EDLIN, you can call out specific lines (by number) for editing.

The /N switch may be used alone, so that lines containing matches are displayed with their relative line numbers, or with the /V switch, so that lines that do *not* contain a match are displayed with their relative line numbers.

/V—Variant Lines

Normally, FIND displays all lines from the specified file(s) that contain a series of characters matching the specified string. Use the /V switch to display all lines (and only those lines) that do *not* contain a match. If the /N switch is also used, each line is preceded by its relative line number.

Locating a String in Text Files

Suppose you have a disk in drive B that contains three research paper files named PAPER1.TXT, PAPER2.TXT, and PAPER3.TXT. Further, suppose that some or all of the files may contain erroneous references to *Waiting for Godot*. To locate all the references to this play, you could enter

```
FIND /N "Waiting for Godot" B:PAPER1.TXTB:PAPER2.TXT B:PAPER3.TXT
```

and press **RETURN**. The /N switch will cause each line displayed to be preceded by its relative line number within the file. Notice that the string you wish to locate must be enclosed in quotation marks.

When you press RETURN, FIND will begin searching the files you specified in the order in which they appear on the command line. The screen will display information in the format

```
---------- B:PAPER1.TXT
[nn]Txttxt txt txt Waiting for Godot txt txt txt
[nn]Waiting for Godot txttxttxttxt txt txttxt xt

---------- B:PAPER2.TXT
            [nn]txttxttxttxt stuff and txt Waiting for Godot txt

---------- B:PAPER3.TXT

A>
```

where [*nn*] is the relative line number of the adjacent line.

If no lines are displayed following the FIND header for a specified file (as for B:PAPER3.TXT in the above example), this means that no match with the specified string was found in the file.

After the FIND operation is completed, the system prompt (`A>` in the above example) is displayed again.

Using the information displayed by the FIND command, you may easily access the lines in the files that contain the erroneous references and correct them.

Using the Variant Lines Switch

Suppose you have a disk in drive B that contains program files with various extensions and data files with the extension .DAT, and that you wish to review the directory to see what program files are on the disk. You could display the entire directory and read it, but a simpler way is to display a selective directory for the disk (that is, a directory that does not include the data files). You may do this by using the directory command and piping the output to FIND, as follows

```
DIR B: | FIND /V "DAT"
```

When you enter this command line and press RETURN, the screen will display a directory of all files that do not have the extension .DAT (that is, a directory of all program files). (In this example, | is the pipe that feeds the output of the DIR command to FIND. For more information on pipes, refer to Chapter 8, "Input/Output Features.")

Error Messages

`Incorrect DOS version`

EXPLANATION: FIND will execute only under MS-DOS version 2 or higher. This message will be displayed if you try to invoke the FIND operation using an incompatible version of MS-DOS.

`FIND: File not found` *filespec*

EXPLANATION: The file specified (*filespec*) either does not exist or is not on the disk specified. Check the file you specified to make sure you entered the correct file specification and reenter the FIND command for the desired file.

`FIND: Invalid number of parameters`

EXPLANATION: You entered the FIND command without specifying the required string. Reenter the FIND command, specifying the string for which you wish FIND to search.

`FIND: Invalid Parameter` *x*

EXPLANATION: You specified an invalid switch (/*x*). When this occurs, FIND will display this error message and then execute as though the invalid switch had not been entered. Any valid switch that was also entered will affect FIND execution as expected.

`FIND: Read error in` *filespec*

EXPLANATION: An error occurred when FIND tried to read the file (*filespec*) specified in the FIND command line.

`FIND: Syntax error`

EXPLANATION: You included an illegal string in the FIND command line. Reenter the command, making certain that the string you specify is properly enclosed in quotation marks.

FOR (Resident, Batch-Processing)

Purpose

The FOR command is used to repeat an MS-DOS command during batch file and interactive file processing.

Entry Forms

```
FOR %%variable IN (set) DO command
FOR %variable IN (set) DO command
```

where ***variable*** is a replaceable parameter which can be any character except 0,1,2,3,...9—to avoid confusion with the %0-%9 batch parameters;
IN is an operator under the FOR command that instructs FOR as to what (*set*) to perform the *command* on.
(***set***) is one of the following:
  (***filespec***...);
  (***pathname***...);
  (***filename***...);
  (**ASCII *character string*...**); or

  (***afn***), where *afn* is any ambiguous file name that uses wildcard characters such as * and/or ?, and where there may be only *afn* included in (*set*);
DO is an operator under the FOR command that tells FOR what *command* to execute.
command is any valid MS-DOS command.

Preliminary Concepts

The resident, batch-processing commands are most often executed from within a batch file, although they may be used directly from the command line in some instances. Refer to Chapter 5, "Command Features," for a complete explanation of batch file creation and execution.

NOTE: You must always end each line of a batch file by pressing the RETURN key.

The %%*variable* is assigned sequentially to each member of (*set*), and then *command* is executed for each member of (*set*). If a member of *set* is an expression involving the wildcard characters, * and/or ?, then the variable is set to each matching pattern (which may be a file name) from disk. In this case of an ambiguous file name, (*set*) may include only *one* member, and any member of (*set*) besides the first will be ignored.

NOTE: If (*set*) is an ambiguous file name (afn), it cannot also be a path name.

Command Line Entry

The FOR command is entered, followed by the parameters listed below:

The *variable* parameter can be any character except 0,1,2,3,..,9—to avoid confusion with the %0-%9 replaceable parameters.

The (*set*) parameter can be any of the parameters listed under Entry Forms and may also be a list of those parameters. When using a list within (*set*), you must separate each member with a space, and you cannot exceed a total line length of 128 characters.

NOTE: The (*set*) parameter must be enclosed in parentheses.

The %%*variable* is set sequentially to each member of (*set*). If a member of (*set*) is an expression containing * and/or ?, then *variable* is set to each matching pattern on disk.

The *command* parameter can be any valid MS-DOS command.

Usage for Interactive Processing

If you wanted to display the directories of a number of programs, you would enter the command

```
FOR %f IN (programx.asm programy.asm programz.asm) DO DIR %f
```

and press the **RETURN** key. The screen would then display the directories of the following three programs, in sequence:

```
programx.asm
programy.asm
programz.asm
```

Usage for Batch File Processing

If you wanted to create a batch file that would execute the PRINT command for every file in the directory that has the .LST extension, you would include the following command line in your batch file:

```
FOR %%f IN (*.LST) DO PRINT %%f
```

The '%%' is needed so that after batch parameter processing is done, there is one '%' left. If only '%f' were there, the system processor would see the '%', look at 'f', decide that '%f' was an error (bad parameter reference) and throw out the '%f', so that the command FOR would never see it. If the FOR is *not* in a batch file, then only one '%' should be used (as in the example under Interactive Processing.

NOTE: Only one FOR command may be specified per command line.

Error Messages

```
Invalid number of parameters
```

EXPLANATION: This message will be displayed if you have left out one of the command parameters, or have tried to enter more than one FOR command per command line. Reenter the command.

```
Syntax error
```

EXPLANATION: This message will be displayed if you have not followed the specific syntax rules for entering this command. Reenter the command line.

FORMAT (Transient)

Purpose

The FORMAT (Format Disk) command prepares a disk to accept MS-DOS files by writing the file allocation information on it that MS-DOS uses to locate data on the disk.

Entry Form

```
FORMAT [d:] [/x ...]
```

where ***d*** is the name of the drive that contains the disk you want formatted; and

/x... is any combination of the following switches:

/C This switch causes FORMAT *not* to initialize the disk, as it normally would, but rather to simply clear the directory and file allocation tables. Note that you must also specify the /M switch, if the disk was originally formatted by specifying the /M switch.

/8	This switch causes FORMAT to use the 8 sectors per track format, instead of the 9 sectors per track format, which is the default.
/M	This switch tells FORMAT to initialize the disk as a single- sided disk.
/N	This switch causes FORMAT to suppress the onscreen prompts. This is useful when the FORMAT utility is used within a batch file.
/S	This switch causes FORMAT to copy the system files from the disk in the default drive to the newly formatted disk.
/V	This switch causes FORMAT to check the disk after formatting to verify that it was formatted correctly.

Preliminary Concepts

FORMAT is one of the most fundamental utilities in MS-DOS. The information that it records on the surface of the disk is used by MS-DOS to determine where to read or write information. It allows you the option of single or double-sided disks and places system files on the disks that you want to make bootable. FORMAT is the first program that you run on any new disk.

The FORMAT utility initializes the directory and File Allocation Tables. If no drive is specified, FORMAT will prompt you for a drive name. The boot loader is copied onto track 0, starting at sector 0. (This occurs whether or not the /S switch is specified.)

If the disk's format were visible, it would appear as:

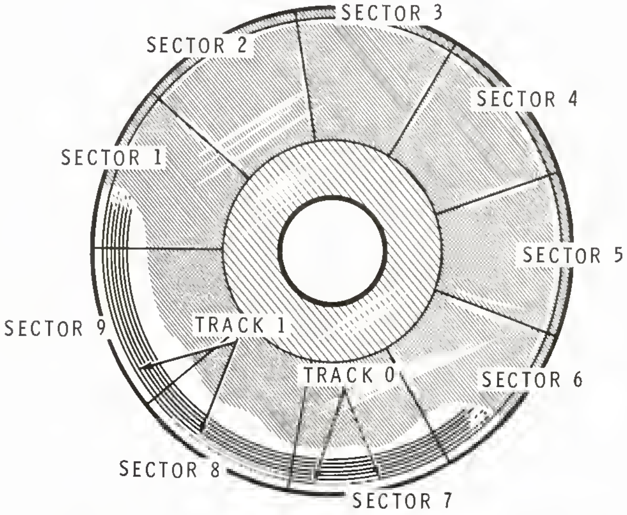


Figure 11.11. Layout of Disk Sectors and Tracks

The number of tracks per disk side depends on the type of disk being used. Your microcomputer uses floppy disks that are 5.25-inch, with 48-tpi, and have 40 tracks per side.

Different types of disks require different formats. FORMAT assumes that the disk is to be formatted on both sides. If the disk needs to be formatted on only one side, FORMAT is told to do so with the /M switch.

Command Line Entry

FORMAT may be entered at the system prompt with a variety of options, or FORMAT may be entered without selecting any options. If no options are entered, FORMAT assumes the default values:

- double-sided disk,
- 9 sectors per track,
- no system transferred, and
- the screen will prompt for all entries.

The FORMAT command line without options is entered as

```
A>FORMAT
```

and pressing **RETURN**. When this is entered, FORMAT responds with:

```
                FORMAT version 2.0


Drive to format? _:
```

When you enter the drive name, the screen display changes. For example, if you entered **B** at the previous prompt the screen would display:

```
Insert new disk in drive B
and press RETURN when ready.
```

You can also specify the drive you want to format in the command line. For example, if you wanted to format a disk in drive B, you would enter

```
A>FORMAT B:
```

and press **RETURN**. This will cause the following to display:

```
                FORMAT version 2.0


Insert new disk in drive B
and press RETURN when ready.
```

You would place the disk to be formatted in drive B, and press **RETURN**.

In addition, several options are available in the form of switches:

/C The /C switch causes FORMAT not to initialize the disk, but rather to simply clear the directory and File Allocation Tables. Note that you must use the /M switch if the disk was originally formatted with the /M switch specified.

/8 This switch causes FORMAT to use the 8 sectors per track format (used by releases of MS-DOS prior to version 2) instead of the 9 sectors per track format, which is the current default.

/M The /M switch tells FORMAT to initialize the disk as a single-sided disk. This involves recording data on only one side of the disk.

/N The /N switch causes FORMAT to suppress prompts for the insertion of a disk and not to report any statistics regarding the formatting process. This is useful when the FORMAT program is used in a batch file.

/S The /S switch causes FORMAT to copy the system files from the disk in the default drive (usually the booted disk) to the newly formatted disk. These files are:

IO.SYS (hidden file)
MSDOS.SYS (hidden file)
COMMAND.COM

NOTE: These three files are copied in the order shown.

/V The /V switch causes FORMAT to check the disk after formatting to verify that it was formatted correctly. Any bad sectors that are found are marked in the File Allocation Table and will never be allocated for your data. If you are formatting a Winchester disk partition, the bad sectors located by the PREP utility will be marked in the File Allocation Table, whether you use the /V option or not. At the end of the format operation, if bad sectors are found, the byte count for the bad sectors found is displayed in a message. This switch also instructs FORMAT to mark any bad sectors it finds on the disk, so that they cannot be used.

Completion Messages

When FORMAT has finished formatting a disk, it displays a short message that briefly reports what it has done and then asks if you have more disks to format. Some examples of FORMAT's completion messages are

for 5.25-inch (48-tpi) with no switches selected:

```
362496 bytes total disk space
362496 bytes available on disk
```

for 5.25-inch (48-tpi) with the /S switch selected:

```
362496 bytes total disk space
 45568 bytes used by system
315392 bytes available on disk
```

for 5.25-inch (48-tpi) with the /M switch:

```
179712 bytes total disk space
179712 bytes available on disk
```

for 5.25-inch (48-tpi) with both /M and /S switches:

```
179712 bytes total disk space
 45568 bytes used by system
134144 bytes available on disk
```

All of the completion messages in FORMAT take this form, though the byte count given may vary somewhat.

Immediately after the completion message, FORMAT asks:

```
Do you wish to format another disk (Y/N)?
```

Pressing **N** for no returns you to the system prompt. Responding with a **Y** for yes will cause the program to run again (using the same options previously selected).

If you are formatting a floppy disk, the message that appears after you specify the drive to be formatted is:

```
Insert new disk in drive d
and press RETURN when ready.
```

where *d* is the designated drive.

If you are formatting a partition of a Winchester hard disk, as opposed to a floppy disk, the message displayed will be:

```
Will format partition assigned drive d
Press RETURN when ready.
```

If you specify the /S switch, the message:

```
System transferred
```

will be displayed after the system is placed on the specified disk.

If you have removed your working MS-DOS disk from the default drive before a formatting operation is completed, the following message will be displayed:

```
Insert DOS disk in drive d
and press any key when ready.
```

FORMAT will prompt you to enter a volume label after a disk has been formatted with the following display:

```
Enter desired volume label (11 characters, RETURN for none)?
```

If you wish to add a volume label to the disk being formatted, you should do so now. It may be up to 11 characters long and can use any of the valid file name characters described in Chapter 1, "Beginning Concepts." You should note that volume labels can *only* be entered at the time a disk is formatted.

Advanced Concepts

All MS-DOS disk formats have an essential structure in common:

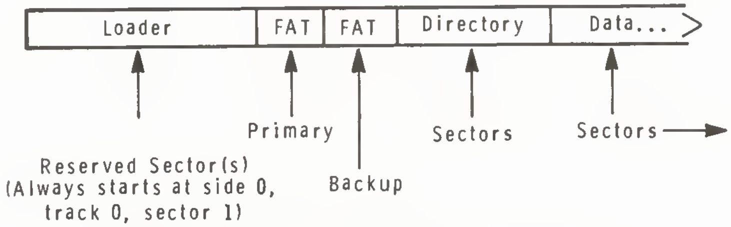


Figure 11.12. Basic Primary Sector Structure for Initialized Disks

This figure shows how the information on a disk would appear to be organized if you could see it all in consecutive order.

For all options selected, except the /C switch and when formatting partitions of Winchester hard disks, FORMAT first initializes each track on the disk. It next goes back to sector 1, track 0, side 0, and writes the Boot Loader code to the first 512 data byte locations on the disk.

FORMAT next writes one dummy FAT. This is different for each type of disk format; the various formats are shown below:

Table 11.2. FAT Values in Hexadecimal Notation

FAT VALUE (IN HEX)	DISK FORMAT
FFH	48-tpi double-sided, 8 sectors per track
FEH	48-tpi single-sided, 8 sectors per track
FDH	48-tpi double-sided, 9 sectors per track
FCH	48-tpi single-sided, 9 sectors per track

The space allocated for the directory and data sectors varies with the type of disk format. In Figures 11.13 through 11.16 the different sector allocations for the initial disk sectors on side 0, track 0, sector 1 (up to their first data sector) are shown for each of the current MS-DOS formats.

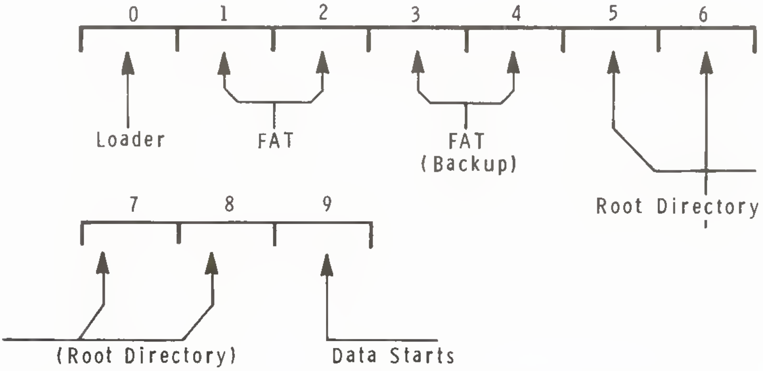


Figure 11.13. **5.25-inch, Single-sided, 48-tpi Disk Format, No Switches Specified (9 Sectors/Track)**

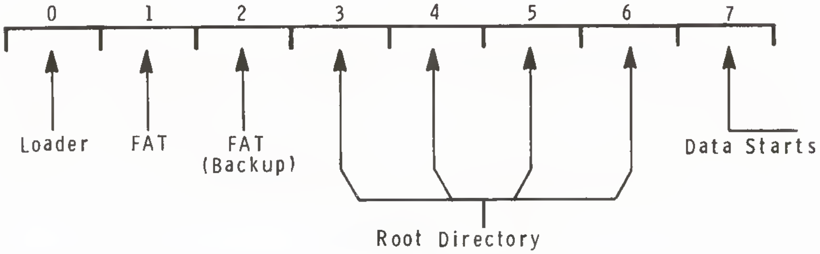


Figure 11.14. **5.25-inch, Single-sided, 48-tpi Disk Format, with the /8 Switch Specified (8 Sectors/Track)**

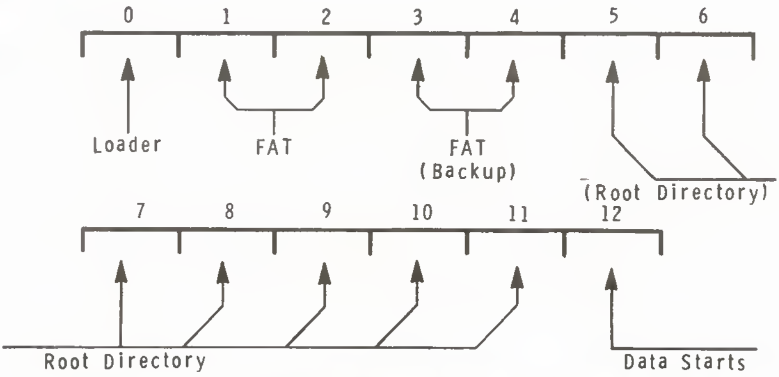


Figure 11.15. 5.25-inch, Double-sided, 48-tpi Disk Format, with No Switches Specified (9 Sectors/Track)

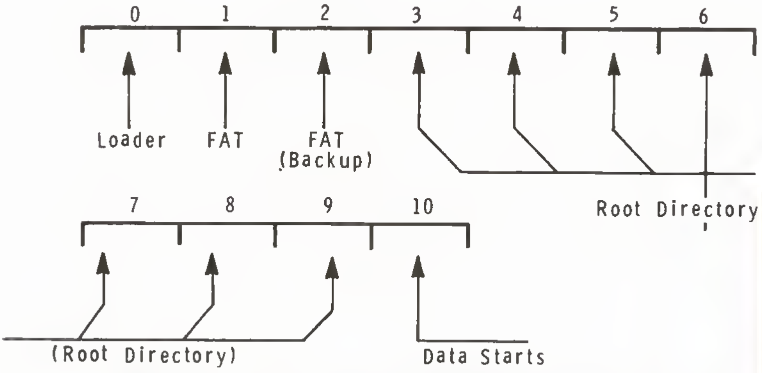


Figure 11.16. 5.25-inch, Double-sided, 48-tpi Disk Format, with the /8 Switch Specified (8 Sectors/Track)

Disk Format

The clusters (data groupings) are arranged on disk to minimize head movement for multi-sided media. All of the space on a track (or cylinder, in the case of a Winchester hard disk) is allocated before moving on to the next track.

The first byte of the File Allocation Table contains an entry specific to the type of media. For example, a double-sided, 9 sectors per track, disk would have a hexadecimal FD for the first byte.

The second and third bytes always contain FFFFH.

The third FAT entry begins the mapping of the data area (cluster 002). Files in the data area are not necessarily written sequentially on the disk. The data area space is allocated one cluster at a time, skipping over clusters already allocated. The first free cluster found will be the next cluster allocated, regardless of its physical location on the disk. This permits the most efficient utilization of disk space because clusters made available by erasing files can be allocated for new files.

MS-DOS Disk Directory

FORMAT initially builds the root directory for all disks.

Since directories other than the root directory are actually files, the only limit to the number of entries is the memory capacity of the disk.

All directory entries are 32 bytes in length and are in the following format (byte offsets are in decimal):

0-7 File name. The first byte of this field indicates its status.

00H Never been used. This is used to limit the length of directory searches, for performance reasons.

E5H Was used, but the file has been erased.

2EH The entry is for this directory. If the second byte is also 2EH, then the cluster field contains the cluster number of this directory's parent directory (0000H, if the parent directory is root directory).

Any other character is the first character of a file name.

8-10 File name extension

11 File attribute. The attribute byte is mapped as follows (values are in hexadecimal):

01 File is marked read-only. An attempt to open the file for writing, by using function call 4DH, results in an error code being returned. This value can be used along with other values below.

02 Hidden file. The file is excluded from normal directory searches.

04 System file. The file is excluded from normal directory searches.

08 The entry contains the volume label in the first 11 bytes. The entry contains no other usable information (except date and time of creation) and may exist only in the root directory.

10 The entry defines a subdirectory, and is excluded from normal directory searches.

20 Archive bit. The bit is set *on* whenever the file has been written to and closed. This bit can be used along with other attribute bits.

NOTE: The system files (IO.SYS and MSDOS.SYS) are marked as *read-only*, *hidden*, and *system* files. Files can be marked *hidden* when they are created.

12-21	Reserved.
22-23	Time the file was created or last updated.
24-25	Date the file was created or last updated.
28-31	File size in bytes. The first word contains the low-order part of the size. Both words are stored with the least significant byte first.

Error Messages

```
Disk unsuitable for system disk
```

EXPLANATION: The disk you wish to format with the operating system (/S switch) is not usable for that purpose. You must use a different disk for this operation.

```
Format Failure
```

EXPLANATION: This message will be displayed if your disk is unformattable because of damage, or because it is not the right type for the formatting command specified.

```
Insufficient memory for system transfer
```

EXPLANATION: Your system is configured with too little memory for this operation, or a portion of your system memory (RAM) is not functioning properly. You probably need to add more memory to your system.

```
Invalid characters in volume ID
```

EXPLANATION: Only valid file name characters are allowed for volume labels. Reenter volume ID by using valid characters. (Refer to Chapter 1, "Introductory Concepts," for a listing of valid characters.)

```
Invalid drive specification
```

EXPLANATION: You designated an invalid drive name; reenter the command with a valid drive name.

```
Invalid parameters
```

EXPLANATION: The command line contained invalid characters or used valid characters in an invalid way. You must reenter the command.

```
Track 0 bad - disk unusable
```

EXPLANATION: The disk you are using for this operation is bad; you must use another.

GOTO (Resident, Batch-Processing)

Purpose

The GOTO command is used to branch unconditionally out of the normal execution sequence in a batch file to a specified label within the batch file.

Entry Form

`GOTO` ***label***

where ***label*** is a reference to a character string definition, located elsewhere within the batch file. The label *definition* must be preceded by a colon (:). Its first eight characters must be significant.

Preliminary Concepts

The resident, batch-processing commands are most often executed from within a batch file, although they may be used directly from the command line in some instances. Refer to Chapter 5, "Command Features," for a complete explanation of batch file creation and execution.

NOTE: You must always end each line within a batch file by pressing the RETURN key.

GOTO transfers control to the line immediately below the :*label*. If no label has been defined, the current batch file will terminate. Only the first eight characters of a label are significant.

The GOTO command will unconditionally transfer control to the line immediately following the one which contains the specified label. To insert a label in a batch file, you must enter a colon (:), followed by the *label*.

Command Line Entry

To enter the GOTO command, type **GOTO**, followed by the label you have defined elsewhere within the batch file.

When the batch processor comes to the GOTO command, it will automatically transfer control to the named *label*.

Executing the GOTO Command Within a Batch File

If you include the lines

```
:foo.
REM looping...
GOTO foo
```

within a batch file, it will produce an infinite sequence of the message:

```
REM looping...
GOTO foo
```

This happens because each time the GOTO command is executed, it transfers control to the `:foo` label which causes the remark, `looping...` to be displayed on your screen.

Starting a line in a batch file with a colon (:) causes the line to be ignored by the batch process. The characters following the colon define a label. This procedure may also be used to put in comment lines, if the label remains undefined.

NOTE: The labels within a batch file are never displayed while the batch file is executing. As in the above example – *:foo*.

Error Message

```
Label not found
```

EXPLANATION: This message will be displayed if the *label* is not defined, or is not present.

IF (Resident, Batch-Processing)

Purpose

The IF command is used to allow conditional execution of MS-DOS commands under batch-processing.

Entry Form

`IF [NOT]` ***`condition command`***

where **NOT** is used to negate the value found by ***condition***;

> ***condition*** is one of the following:
>
> **ERROR LEVEL** ***number***, where ***number*** is the value of the exit code of the program previously executed by the system.
> ***string1 = = string2*** The *condition* will be true if, and only if, *string1* and *string2* are identical.
> **EXIST** ***filespec*** The *condition* will be true if, and only if, *filespec* exists on the designated drive; and
> ***command*** is any valid MS-DOS command.

Preliminary Concepts

The resident, batch-processing commands are most often executed from within a batch file, although they may be used directly from the command line in some instances. Refer to Chapter 5, "Command Features," for a complete explanation of batch file creation and execution.

NOTE: You must always end each line within a batch file by pressing the RETURN key.

The IF statement allows conditional execution of commands. When the *condition* is true, the *command* is executed. Otherwise, the *command* is ignored.

If a *condition* is true, the batch file will execute the *command*; otherwise it will be skipped. The condition can compare two strings, check to see if a file exists on the disk, or check for an ERROR LEVEL number. The *command* portion of the IF statement may include a GOTO statement.

GOTO labels must begin with a colon (:), and the GOTO statement will execute the command on the line below the GOTO label. Refer to the section on the GOTO command in this chapter for more information on its use.

You may also use replaceable parameters within the IF statement, as variables to be replaced with specific values at the time of execution.

The replaceable parameters %0 through %9 are used within a batch file as "dummy" parameters which will be replaced sequentially with real values when the batch file is executed.

The parameters are substituted, in order, on the command line. A one-to-one correspondence is set up between the prototype commands in the command line and the replaceable parameters in the batch file. That is, '%1' stands for the first value (name, number, or text) typed after the command, and '%2' stands for the second value typed after the command, and so on. This relationship is illustrated in the following example.

The example command line below corresponds on a one-to-one basis with the replaceable parameters shown under each component of the command line:

```
batch filename   filespec1   filespec2   filespec3 ...      filespec9
|                |           |           |                  |
|                |           |           |                  |
%0               %1          %2          %3 ...             %9
```

NOTE: The replaceable parameter '%0' is always replaced by the drive name (if specified), and the file name of the batch file.

Although you may only use up to ten replaceable parameters within a batch file (%0 through %9), you can avoid this limitation by using the SHIFT command. The SHIFT command shifts the parameters to the left, one parameter at a time. The SHIFT command is described in detail later in this chapter.

Command Line Entry

The IF command uses the *condition* parameter to check three different types of conditions that must be met before the complete IF statement can be executed.

These three types of conditions are:

1. **ERROR LEVEL number**

   The *condition* parameter is true if, and only if, the previous program executed by the system had an exit code of *number* or higher. (*number* is specified as a binary value.)

   NOTE: Most MS-DOS programs currently return an exit code of 0 at all times.

2. The *string1==string2* parameter is true if, and only if, *string1* and *string2* are identical (after parameter substitution; if substitution is used).

   Strings may not have embedded separators (such as commas or spaces).

3. The **IF EXIST** parameter checks to see if a file exists on the designated disk. If the file does exist, the EXIST parameter is true and the batch file executes the specified *command* included in the IF statement.

For example, you may wish to include the following commands in a batch file to check that a file exists:

```
IF EXIST %1 GOTO X
GOTO Y
:X
ECHO THAT FILE IS ON DISK
:Y
```

These would enable you to specify a file name and have your screen display the message

```
THAT FILE IS ON DISK
```

if the specified file did exist on that disk.

The following IF statement offers another example of how this works:

```
IF EXIST filespec command
```

The **IF EXIST** statement would be true if, and only if, *filespec* does indeed exist on the designated drive. Path names cannot be used with the *filespec* variable in this situation.

The following examples illustrate how this works when the parameter [NOT] precedes the *condition* parameter:

```
IF NOT condition
```

The IF statement would be true if, and only if, the *condition* parameter is false.

The following example illustrates the **IF EXIST** parameter along with the GOTO command:

```
IF EXIST FILEFORM.DOC GOTO XYZ
-
-
-
:XYZ
command
```

This command will cause the processor to go to the label **:XYZ** to execute the *command*, if **FILEFORM.DOC** is found on the default drive. If **FILEFORM.DOC** is not found, **GOTO XYZ** will not be executed. Instead, processing would continue with the next command in sequence within the batch file.

On the other hand, if you specify

```
IF NOT EXIST FILEFORM.DOC GOTO XYZ
-
-
-
:XYZ
command
```

the condition of **FILEFORM.DOC** *not* being on the default drive would have to be met before the command **GOTO XYZ** could be executed.

If **FILEFORM.DOC** *is* on the default drive, processing would *not* execute **GOTO XYZ**. Instead, processing would continue with the next command in the batch file.

The IF command using the *string1 = = string2* parameter is illustrated in the next example:

```
IF %1==Hello ECHO How are you.
```

If a batch file is executed with this command, where **Hello** is given as the %1 parameter, it would make the condition true. The ECHO command would be executed, causing the screen to display:

```
How are you.
```

If **Hello** is not given as the %1 parameter, the condition would not be true. The ECHO command would not be executed and processing would continue with the next command in the batch file.

KEYB*xxxx* (Transient)

Purpose

Loads a foreign language keyboard layout into memory.

Entry Form

```
[d:]KEYBxxxx
```

where ***d*:** is the name of the drive that contains the transient command file; and
xxxx is the code for the keyboard layout. The codes are listed in Table 11.2a.

Preliminary Concepts

The keyboard installed with the KEYB*xxxx* command replaces the ROM BIOS keyboard program in memory. When a keyboard program is loaded, it is allocated approximately 2K of memory.

You can keyboard format without resetting the system since any new invocation of the KEYB*xxxx* command will replace the previous installation.

Hold down CTRL-ALT and press F1 to change from any foreign keyboard layout to the United States format. The layout for the United States keyboard is the default. To reinstate the foreign keyboard layout, hold down CTRL-ALT and press F2.

KEYB*xxxx* Character Sets

There are two character sets that can be used for foreign language keyboard layouts. The first and most commonly used is the 256byte IBM-compatible, which includes graphics characters (128—256) in ROM. Most software commercially available will support the use of the 256-byte character set.

The other character set you can access is the standard ASCII that uses only 128 characters, none of which are graphic. This standard set will probably only be chosen when a printer designed for use with international ASCII only is in operation.

Table 11.2a. KEYB*xxxx* Codes

LANGUAGE	256-BYTE CODE	128-BYTE CODE
Australian	US	AUS
Danish	DA	ADA
Dutch	US	AUS
English, United Kingdom	UK	AUK
English, United States	US	AUS
Finnish	SW	ASW
French	FR	AFR
French-Canadian, Electronic	CANE	none
French-Canadian, Selectric	CANS	none
German	GR	AGR
Italian	IT	AIT
Norwegian	NO	ANO
Spanish	SP	ASP
Swedish	SW	ASW
Swiss/French	CHF	none
Swiss/German	CHG	none

NOTE: Keyboard codes are not provided for all the countries listed. Use the code for the keyboard characters that most closely resemble the language you will be using. The most commonly used Keyboard codes are the 256 byte extended character set codes. Most software commercially available for your microcomputer will support the use of the 256 byte codes. The ASCII standard character set will be used if you have a printer designed for use with International ASCII only.

Table 11.2a lists the KEYB*xxxx* codes for the 256-byte character set and the standard 128-byte ASCII character set. Choose one for the KEYB*xxxx* command line entry. If there is no code for the language you will be using, choose the most similar one.

Error Messages

`Internal table sizes do not match`

EXPLANATION: The tables that define the keyboard map you have selected have been incorrectly altered. You cannot use the selected keyboard when you see this message.

`Unable to load Key map routine`

EXPLANATION: The ROM keyboard routine has already been replaced by a keyboard routine other than a KEYB*xxxx* type. In order to run your current selection, you must reset the system and install your choice.

MAP (Transient)

Purpose

Used for drive-specific application programs to instruct the operating system to use a different drive from that specified by the application for disk operations.

Entry Forms

```
MAP ?
MAP [x=y [...]]
```

where **?** invokes the MAP help screen display;
x is the program-specified drive you wish to remap; and
y is the drive to which you want logical drive x mapped.

Preliminary Concepts

The MAP command enables you to temporarily reassign (remap) logical drive names to different physical devices. This command makes it more convenient to use applications that are designed to perform disk operations using specific drives and do not allow the user to specify the drive(s) to be used.

For example, some application programs are designed such that program files must be on the disk in drive A and data files are read from and written to the disk in drive B. For such cases, MAP allows you to remap drive A and/or drive B so that the system actually uses other drives. That is, if you remapped drive A to drive E, the system would access drive E every time that drive A was called by the application. MAP may be used for Winchester disk partitions as well as for floppy disk drives.

IMPORTANT: MAP should *not* be used to remap drives for normal MS-DOS operations. This is because reassignment of drive names through MAP can hide the true device type and characteristics from system commands and programs that require actual device information. Because the following MS-DOS utilities ignore drive remapping, they should *never* be used on a mapped drive:

ASSIGN
BACKUP with formatting
DISKCOMP
DISKCOPY
FORMAT
RDCPM

When you use MAP to reassign drive names so that you can more conveniently run a drive-specific application, you should restore normal drive assignments when you are finished with the application. (Refer to Command Line Entry in this section.) If you develop an application program, it is recommended that you avoid using specific drive assignments in your program and allow the user to specify the drive(s) to be used during program execution.

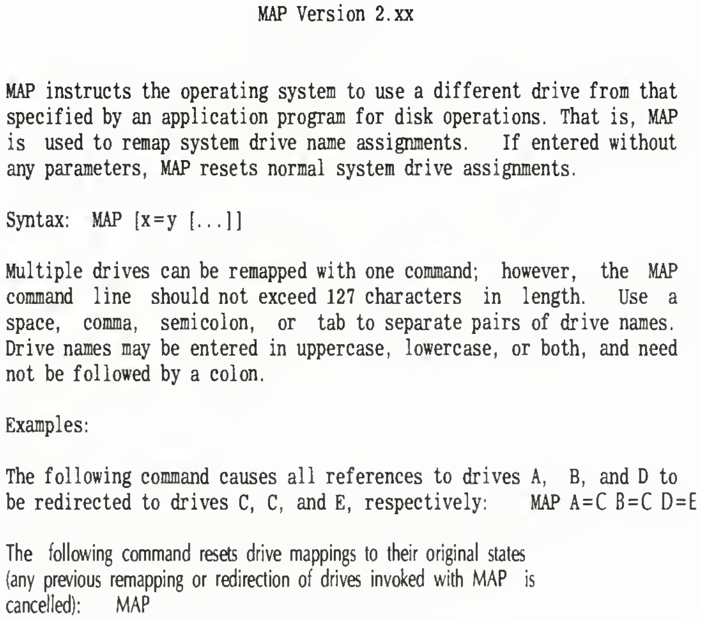
The drives you remap with MAP remain in system memory for all operations until you redefine them with another MAP command, reset normal assignments by entering a MAP command without the optional parameters, or reboot your system.

MAP Help Screen

To display the MAP help screen, enter:

```
MAP ?
```

at the system prompt and press **RETURN**. The screen will display a summary of MAP usage and valid command line entry forms as shown in Figure 11.16a. The help screen is followed by the system prompt, making it easy for you to refer to the information that is provided as you enter a MAP command line.



```
                    MAP Version 2.xx


MAP instructs the operating system to use a different drive from that
specified by an application program for disk operations. That is, MAP
is  used to remap system drive name assignments.   If entered without
any parameters, MAP resets normal system drive assignments.

Syntax:  MAP [x=y [...]]

Multiple drives can be remapped with one command;  however,  the  MAP
command  line  should not exceed 127 characters  in  length.   Use  a
space,  comma,  semicolon,  or  tab to separate pairs of drive names.
Drive names may be entered in uppercase, lowercase, or both, and need
not be followed by a colon.

Examples:

The following command causes all references to drives A,  B, and D to
be redirected to drives C, C, and E, respectively:    MAP A=C B=C D=E

The  following command resets drive mappings to their original states
(any previous remapping or redirection of drives invoked with MAP  is
cancelled):    MAP
```

Figure 11.16a. MAP Help Screen

Command Line Entry

To invoke MAP, you must enter the command name (MAP) followed by parameters specifying the drive to be remapped and the drive to which you want it mapped. The command name and drive parameters must be separated by a space or equivalent MS-DOS delimiter, and the drive parameters must be separated by an equal sign (=). Thus, to remap one drive to another, enter a command line in the form:

```
MAP x=y
```

where **x** is the drive you wish to remap (the drive specified for use by the application), and
y is the drive to which you wish to map drive *x*.

For example, if you are going to run an application that requires that program files be on the disk in drive A and you wish to have the system access drive C instead, you could enter the following command at the system prompt and press **RETURN** before invoking the application:

```
MAP A=C
```

Once you have remapped drive A to drive C in this way, the system will actually access the disk in drive C whenever it is directed by the application to access drive A.

Notice that you must specify both the original (application-specific) drive name and the drive that you want the system to access. You do not, however, have to enter a colon (:) as part of either drive name. The name of the drive you are going to remap (*x*) must be entered first and the name of the drive you want the system to access instead (*y*) is entered immediately following the equal sign. You may not include any spaces or delimiters other than the equal sign between the two drive names.

You may remap more than one drive by entering a single MAP command line. To do this, enter more than one *x*=*y* parameter string following the command name, separating each string from adjacent ones by a space or equivalent MS-DOS delimiter.

Thus, if you wished to remap drive A to drive C and drive B to drive D, you could do so by entering

```
MAP A=C B=D
```

at the system prompt and pressing **RETURN**. You can remap any number of drives that you wish, as long as the command line as a whole does not exceed the input buffer limit of 127 characters.

To restore normal system drive name assignments such that they are the same as those that exist upon bootup, enter:

```
MAP
```

at the system prompt and press **RETURN**.

Error Messages

```
Incorrect DOS version
```

EXPLANATION: This message will be displayed if you booted up your system with a version of MS-DOS previous to version 2. In order for you to use MAP, MS-DOS version 2 or higher must be resident in memory.

```
Invalid parameter
```

EXPLANATION: This message may be displayed if you entered too many or too few parameters as part of the MAP command line, or if you made a syntax error when you invoked the command or if an attempt is made to map an invalid drive. Reenter a valid MAP command line.

MDISK.DVD RAM-Disk Driver

Purpose

MDISK.DVD is one of two functional examples of user-loadable device drivers that are provided on your MS-DOS version 2 distribution disks. The other device driver is ANSI.SYS. While MDISK and ANSI are *not* commands or utilities that are invoked from the system prompt, they are described in this chapter because they may be loaded and used by user option.

MDISK.DVD is a fully-functional block device driver that enables the system to use machine memory in the same way that floppy disk or Winchester disk storage media is used. While a RAM-disk does not provide permanent storage like disk media (RAM is cleared when you reset or reboot your system), it offers the advantage of much faster access time and thus speeds system operation.

The MDISK driver can be added to the system's linked list of standard device drivers (COM1:, LPT1:, and so on, as described in Chapter 8, "Input/Output Features") by inserting a DEVICE= command in the CONFIG.SYS file in the root directory. Note that this is the *only* way that MDISK can be installed and used; it is loaded into memory at bootup only if it is specified in the CONFIG.SYS file. For more information about the CONFIG.SYS file, refer to Chapter 9, "System Component Features."

Installing MDISK.DVD

To load (install) the MDISK driver, you must include a command line in the following form in a CONFIG.SYS file in the root directory:

```
DEVICE=MDISK.DVD [SIZE=nnn] [START= xxxx]
```

where ***nnn*** is a decimal value from the range of 32 through 640 that specifies the amount of memory, in kilobytes, that you wish the RAM-disk to occupy; and
xxxx is a hexadecimal number of up to four digits that specifies the starting paragraph for the RAM-disk's location in memory.

The SIZE and START parameters must be separated from other parameters in the command line by a space or equivalent MS-DOS delimiter. Note, however, that there should be *no* spaces within any one of the three command line parameters.

If you do not specify the SIZE parameter, the default value for RAM-disk size is 32K bytes. If you do not specify the START parameter for the paragraph (memory address) at which MDISK is to be installed, the operating system determines the location to which the driver data is written. Normally, MS-DOS will put MDISK data in low memory with other device driver data. More information about each of these parameters is provided below.

MDISK SIZE and Directory Entries

The SIZE that you specify for MDISK will be the *approximate* RAM-disk storage capacity. When MDISK is installed, some of the memory allocated for the RAM-disk is reserved for use by the operating system (for directory entries, and so on) just as it is on disk media when you run FORMAT. Thus, the actual storage capacity of MDISK will be slightly less than the specified size.

The maximum number of root directory entries allowed on MDISK is dependent upon the size of MDISK. The larger MDISK is, the more directory entries it can contain. The directory entries supported are shown in the listing below:

MDISK SIZE	MAXIMUM DIRECTORY ENTRIES
32 through 63K	16
64 through 127K	32
128 through 255K	64
256 through 511K	128
512 through 640K	256

MDISK START Address

While the START parameter is not required in the CONFIG.SYS DEVICE= command line, it is useful for installing MDISK in noncontiguous memory that is not normally used by the system.

The address entered as part of the START parameter is a hardware identifier for the paragraph (memory address) at which you want MDISK installed. However, all RAM is not operationally available for MDISK; some areas of memory are required by the operating system.

For this reason, when you do specify a starting address for MDISK, *make sure* that the address you specify is in an area of memory *not* required by the operating system. Otherwise (if you enter an address for memory normally used by MS-DOS), part of MS-DOS will be overwritten by MDISK and results will be unpredictable.

There is no error checking for the actual address you specify, only for verification that the value you enter is a valid hexadecimal number. If you are not sure about your system's memory structure and organization, do not enter the START parameter. The system will then determine the starting address for MDISK.

Accessing MDISK

During system operation, MDISK is accessed by drive name (*d*:) just as a floppy disk or Winchester disk partition is accessed. Since drive names in microcomputer systems operated under MS-DOS are dynamic (depending on the number(s) and type(s) of disk drives in the system), MDISK becomes the next available drive name whenever it is loaded.

For example, if you have two floppy disk drives in your system and install MDISK, then the RAM-disk becomes drive C. (Drive names A and B are always reserved for one or two floppy disk drives. Other drive names are dependent upon the drive configuration used in the system. Refer to Chapter 6, "Bootup Features.") If you have two floppy disk drives and a Winchester disk drive, then MDISK becomes drive G. If you have four floppy disk drives and one Winchester disk drive, then MDISK becomes drive I.

When you install MDISK, a message is displayed during the boot sequence to tell you the drive name that has been assigned for MDISK. The message is:

```
MDISK installed as drive d:
```

where *d*: is the drive name by which MDISK may be accessed after its installation and system bootup are successfully completed.

Note that this message does *not* indicate that MDISK has been successfully loaded. If a SIZE, START, or other error condition exists, the corresponding error message will not be displayed until after this informational message appears.

Errors may occur if you specify an invalid size for MDISK, an invalid hexadecimal number as the starting address, or if your system cannot support MDISK in the size you requested. Error messages are described in the following section.

All messages pertaining to MDISK are displayed *during* the boot sequence, before the MS-DOS banner.

In using MDISK, remember that you are making use of volatile memory. Each time you load MDISK at bootup, it is like inserting a newly formatted disk in a drive. If you wish to save any of the contents of MDISK at the end of a work session, you must run BACKUP or copy the file(s) to a floppy disk or Winchester disk partition before you reset or turn off your system.

Error Messages

```
Bad or missing MDISK.DVD
```

EXPLANATION: When this message is displayed during the boot sequence, it means that the MDISK driver was not installed in memory and cannot be used.

It is possible to specify an MDISK size that cannot be supported by your system's current memory configuration even though the size specified is within the allowed range. Such a situation is generally the cause of this error message.

If this occurs, reevaluate available memory, edit the CONFIG.SYS file to reduce the SIZE allocation for MDISK (if possible), and reboot your system. If the error recurs, it may be necessary for you to install additional memory in your system in order for you to load and use MDISK.

```
Invalid MDISK size requested, assigning a size of 32 kilobytes.
```

EXPLANATION: This message is displayed during the boot sequence if an invalid number was specified in the SIZE= parameter of the DEVICE= command in the CONFIG.SYS file. That is, this message is displayed if the specified MDISK size is less than 32 or greater than 640, and thus out of the allowable size range.

Note that when this message is displayed, the MDISK driver has been loaded, but has been allocated only the minimum allowable memory. When this occurs, you may edit the CONFIG.SYS file to specify the desired valid size (if it is more than the 32K bytes allocated by default) and then reboot your system.

```
Invalid MDISK start address requested, address will be ignored.
```

EXPLANATION: This message is displayed during the boot sequence if an invalid number was specified in the START= parameter of the DEVICE= command in the CONFIG.SYS file. While no check is made regarding the actual address specified, the value you enter in the START= parameter must be a valid hexadecimal number of up to four digits.

When this message is displayed, the MDISK driver has been loaded into a memory location determined by the operating system. The START= parameter that was specified is ignored.

MKDIR or MD (Resident)

Purpose

The MKDIR (make directory) command is used to make a new directory.

Entry Forms

```
MKDIR [d:]pathname
MD [d:]pathname
```

where *d* is the letter of the designated disk drive; and

> ***pathname*** is a sequence of characters of the form:
>
> `[\][directory][\directory...]`
>
> Optionally you may use the MS-DOS shorthand notation shown below in lieu of *directory* (as long as they are not the last entry in the path name):
>
> • MS-DOS uses this shorthand symbol to indicate the name of the current (working) directory in all hierarchical directory listings. MS-DOS automatically creates this entry when a directory is made.
>
> •• MS-DOS uses this shorthand symbol to indicate the name of the current directory's parent directory. MS-DOS automatically creates this entry when a directory is made.
>
> **NOTE**: The two shorthand symbols do not exist in the root.

Preliminary Concepts

This command is used to create the hierarchical directory structure. You can create directories by using the MKDIR (or MD) command.

You can make directories anywhere in the tree structure by specifying MKDIR and a path name. MS-DOS will automatically create the • and •• entries in the new directory.

When you create a directory with this command, MS-DOS date stamps the date and time of creation on your disk. This information is also displayed when you execute the DIR command. For more information on date stamping, refer to the sections on DIR, BACKUP, and RESTORE in Chapter 11, "Command Descriptions."

NOTE: Files are not created with this command. Files must be created with EDLIN or applications programs/languages.

Command Line Entry

To create the subdirectory \USER in your root directory, enter

```
MKDIR \USER
```

and press **RETURN**. You can also use the abbreviated form MD, in place of MKDIR.

To create a directory named JOE under \USER, enter

```
MKDIR \USER\JOE
```

and press **RETURN**. If you are not in the root directory, as in the previous example, but are in directory \USER, enter

```
MD JOE
```

and press **RETURN**. This would have the same effect as the previous example.

Error Messages

```
Invalid number of parameters
```

EXPLANATION: This error message will be displayed if you do not enter the proper number of command parameters after MKDIR. For example, if you enter

```
MKDIR
```

and press **RETURN**, MS-DOS will display the above error message. To correct the situation, reenter the command line with the proper number of parameters.

```
Unable to create directory
```

EXPLANATION: This error message will display if you try to use the MKDIR command to create the '•' or '••' directories. For example, if you enter

```
MKDIR .
or
MD ..
```

and press the **RETURN** key, this error message will be displayed. This happens because you cannot create the '.' or '..' directories, only MS-DOS can create these entries.

MODE (Transient)

Purpose

Used to define the mode of operation (protocol) for a display device connected to either the monochrome or the color/graphics card, for a device connected to a serial or parallel communications port, or to remap parallel output for the use of a serial port.

NOTE: Four general methods or options for use of the MODE command are presented in this section. When used to configure a parallel device or a serial device, MODE causes parallel and serial communications intercept code to be made resident in memory. This increases the amount of machine memory required by MS-DOS by approximately 256 bytes.

Entry Forms

```
MODE ?
```

where **?** invokes the MODE help screen display.

```
MODE LPT#: [n] [, [m] [,P]]
```

where *#* is 1, 2, or 3 and designates the specific LPT device to be configured;
n specifies the number of characters per line (80 or 132);
m specifies the number of lines per inch (6 or 8); and
P causes the system to perform continuous retry on time-out errors.

```
MODE [n] [, [m] [, [T]] [,s]]
```

where ***n*** is a value specifying display mode (color, black and white, or monochrome) and width (40 or 80 columns);
m is a value (R or L) used to shift the display right or left;
T causes MODE to produce a test pattern and prompt for display alignment; and
s specifies one of three scrolling modes.

```
MODE COMn:baud[, [parity] [, [databits] [, [stopbits] [,P]]]]
```

where ***n*** is either 1 or 2 and designates the specific COM device to be configured;
baud specifies the baud rate at which your microcomputer will communicate with the device;
parity specifies whether odd, even, or no parity is to be used;
databits specifies word length exclusive of any parity or stop bits;
stopbits is the number of stop bits required; and
P causes the system to perform continuous retry on time-out errors.

```
MODE LPT#: =COMn
```

where *#* is 1, 2, or 3 and designates the specific LPT device to be mapped to a serial port; and
n is either 1 or 2 and designates the specific COM device to which parallel output will be mapped.

Preliminary Concepts

Like CONFIGUR, MODE enables you to configure your operating system to support the use of various input/output devices. While CONFIGUR is menu-driven, MODE enables you to configure your system by entering a single command line. The configuration(s) you define with MODE will remain in memory until you change them or until you reset or reboot your system; they are not saved to disk. Conversely, CONFIGUR gives you the option of saving the configuration you define to disk as well as to memory.

All input and output functions under MS-DOS are accomplished by transferring information between devices. Devices include (but are not limited to) the keyboard, display screen, printers, modems, and disk drives. The part of the operating system that keeps track of the devices connected to your system and that coordinates all transfers of information between devices is the I/O Manager (sometimes called the I/O Handler). For more information about the I/O Manager and other components of the operating system, refer to Chapter 9, "System Component Features."

The I/O Manager includes logical device drivers that define all device characteristics required for the system to identify and use an actual device. Those logical devices that represent the most commonly used device types are configurable for specific physical devices, making it easy for the operating system to use many different devices.

If the operating system could not use a configurable logical device to represent a physical device, either the operating system would need to be rewritten every time a different device was used, or it would be necessary for the operating system to contain information (device drivers) for every possible device. Rewriting an operating system is an extremely long and difficult task. An operating system that contained information for every conceivable device would be extremely long and cumbersome. Thus, configurable logical devices such as those used by MS-DOS present a practical approach that enables the system to work easily with many different devices.

MODE is a command that allows you to configure your system's logical devices so that they reflect the hardware characteristics and protocol requirements of the peripheral equipment (such as a printer, display device, or modem) that you use in your system. *Protocol* refers to the communications method required for successful data transfer between your microcomputer and a peripheral device or other computer. The protocol is also a means of regulating how the peripheral performs its specific operations under the control of the microcomputer; it is a set of standardized signals that the hardware device recognizes and uses that enable its operation to be directed.

MODE Help Screen

A series of MODE help screens are available to provide general information about MODE usage and valid command line entry forms and more detailed information about each of the four configuration options MODE supports.

The first help screen provides a general overview of MODE, and a selection menu that enables you to display more detailed information about each method (option) of using MODE. Each subscreen can be selected from the menu by entering a single digit.

To display the MODE help screen(s), enter:

```
MODE ?
```

at the system prompt and press **RETURN**. The main help screen and menu will be displayed as shown in Figure 11.16b.

```
                         MODE Version 2.xx

MODE  enables  you  to  configure your system for specific peripheral
devices  by  entering a  single  command  line. The  configuration or
protocol that you define with  MODE  remains in memory for use by the
system until you redefine it with  another MODE  command or until you
reset or reboot your system.

Syntax:    MODE ?
           MODE LPT#: [n] [, [m] [,P]]
           MODE  [n] [, [m] [, [T]] [,s]]
           MODE COMn:baud[, [parity] [, [databits] [, [stopbits] [,P]]]]
           MODE LPT#: =COMn

The command lines shown above are used to:
1 -- Display this help screen
2 -- Configure a parallel device
3 -- Configure a color/graphics or monochrome display device
4 -- Configure a serial device
5 -- Remap parallel output to a serial port

For more  information, enter the  number of your  selection. To exit,
press RETURN.
                  Enter selection or press RETURN:
```

Figure 11.16b. MODE Help Screen and Menu

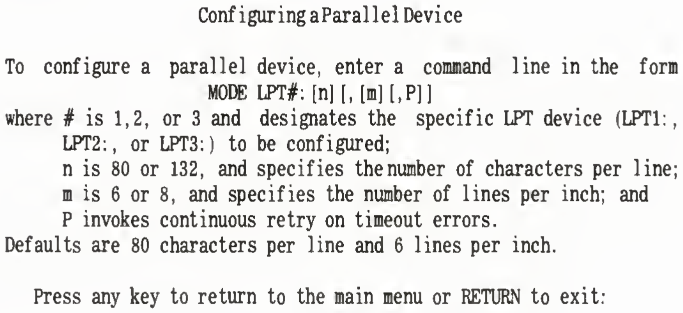
When this screen is displayed, you can exit to the system prompt by pressing **RETURN**. If you wish to display the subscreen for one of the syntax lines/configuration options listed, then enter the single-digit number (1, 2, 3, 4, or 5) corresponding to your choice. Note that if you enter 1, the main help screen and menu will be redisplayed. If you enter any other selection, the appropriate subscreen will be displayed.

The subscreens are shown in Figures 11.16c through 11.16f. All except the Configuring a Display Device subscreen are one screen of information; the Configuring a Display Device help information is on two screens. When the first screen is displayed, you may press any key to display the second screen of information.

The prompt:

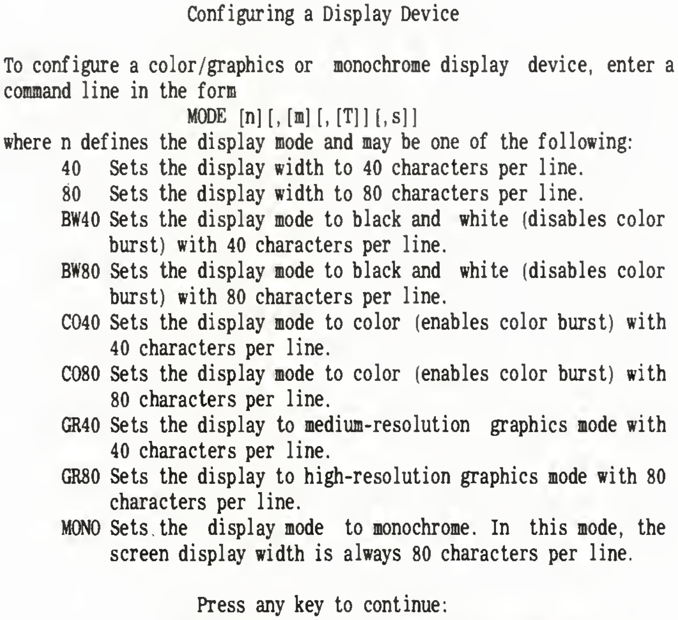
```
Press any key to return to the main menu or RETURN to exit:
```

is at the end of each subscreen. If you wish to exit the help screens, press **RETURN**. The system prompt will be displayed. If you wish to refer back to the main help screen and menu, press any other key. The main help screen will be redisplayed and you may select another subscreen or exit to the system prompt.



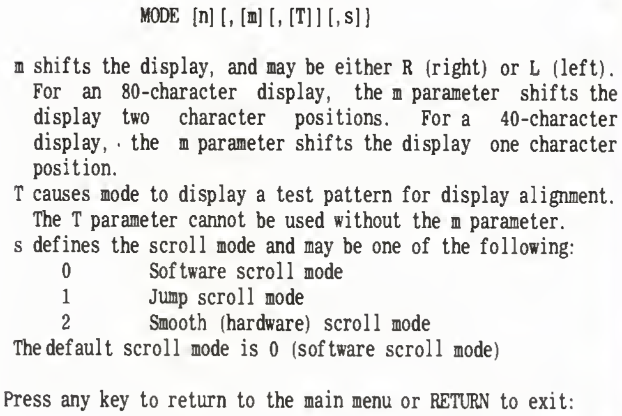
```
                   Configuring a Parallel Device

To  configure a  parallel device, enter a  command  line in the  form
                    MODE LPT#: [n] [, [m] [,P]]
where # is 1,2, or 3 and  designates the  specific LPT device (LPT1:,
      LPT2:, or LPT3:) to be configured;
      n is 80 or 132, and specifies the number of characters per line;
      m is 6 or 8, and specifies the number of lines per inch; and
      P invokes continuous retry on timeout errors.
Defaults are 80 characters per line and 6 lines per inch.

   Press any key to return to the main menu or RETURN to exit:
```

Figure 11.16c. MODE Help Subscreen: Configuring a Parallel Device



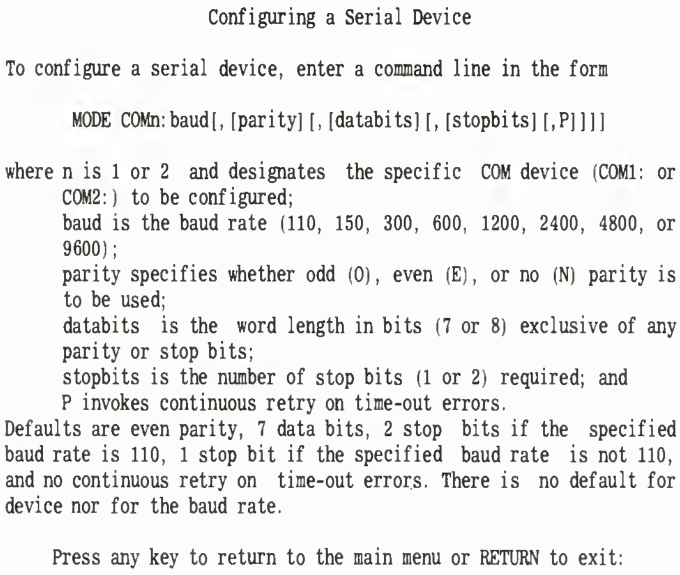
```
                     Configuring a Display Device

To configure a color/graphics or  monochrome display  device, enter a
command line in the form
                     MODE [n][,[m][,[T]][,s]]
where n defines the display mode and may be one of the following:
      40   Sets the display width to 40 characters per line.
      80   Sets the display width to 80 characters per line.
      BW40 Sets the display mode to black and  white (disables color
           burst) with 40 characters per line.
      BW80 Sets the display mode to black and  white (disables color
           burst) with 80 characters per line.
      CO40 Sets the display mode to color (enables color burst) with
           40 characters per line.
      CO80 Sets the display mode to color (enables color burst) with
           80 characters per line.
      GR40 Sets the display to medium-resolution  graphics mode with
           40 characters per line.
      GR80 Sets the display to high-resolution graphics mode with 80
           characters per line.
      MONO Sets the  display mode  to monochrome. In  this mode, the
           screen display width is always 80 characters per line.

                      Press any key to continue:
```

Figure 11.16d-1. MODE Help Subscreen: Configuring a Display Device, Screen 1



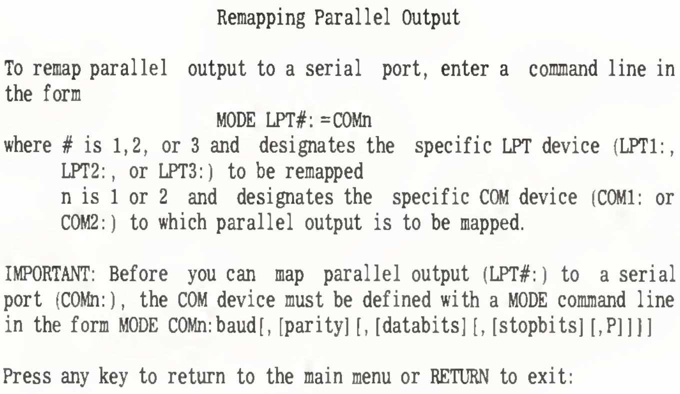
```
                MODE [n][,[m][,[T]][,s]]

     m shifts the display, and may be either R (right) or L (left).
       For  an  80-character  display,  the m parameter  shifts the
       display  two   character   positions.   For a   40-character
       display,  the  m parameter shifts the display  one character
       position.
     T causes mode to display a test pattern for display alignment.
       The T parameter cannot be used without the m parameter.
     s defines the scroll mode and may be one of the following:
          0         Software scroll mode
          1         Jump scroll mode
          2         Smooth (hardware) scroll mode
     The default scroll mode is 0 (software scroll mode)

    Press any key to return to the main menu or RETURN to exit:
```

Figure 11.16d-2. MODE Help Subscreen: Configuring a Display Device, Screen 2



```
                    Configuring a Serial Device

To configure a serial device, enter a command line in the form

      MODE COMn:baud[,[parity][,[databits][,[stopbits][,P]]]]

where n is 1 or 2  and designates  the specific  COM device (COM1: or
      COM2:) to be configured;
      baud is the baud rate (110, 150, 300, 600, 1200, 2400, 4800, or
      9600);
      parity specifies whether odd (O), even (E), or no (N) parity is
      to be used;
      databits  is the  word length in bits (7 or 8) exclusive of any
      parity or stop bits;
      stopbits is the number of stop bits (1 or 2) required; and
      P invokes continuous retry on time-out errors.
Defaults are even parity, 7 data bits, 2 stop  bits if the  specified
baud rate is 110, 1 stop bit if the specified  baud rate  is not 110,
and no continuous retry on  time-out errors. There is  no default for
device nor for the baud rate.

     Press any key to return to the main menu or RETURN to exit:
```

Figure 11.16e. MODE Help Subscreen: Configuring a Serial Device



```
                    Remapping Parallel Output

To remap parallel  output to a serial  port, enter a  command line in
the form
                    MODE LPT#:=COMn
where # is 1,2, or 3 and  designates the  specific LPT device (LPT1:,
      LPT2:, or LPT3:) to be remapped
      n is 1 or 2  and  designates the  specific COM device (COM1: or
      COM2:) to which parallel output is to be mapped.

IMPORTANT: Before  you can  map  parallel output (LPT#:) to  a serial
port (COMn:), the COM device must be defined with a MODE command line
in the form MODE COMn:baud[,[parity][,[databits][,[stopbits][,P]]]]

Press any key to return to the main menu or RETURN to exit:
```

Figure 11.16f. MODE Help Subscreen: Remapping Parallel Output

Command Line Entry

The type of command line you enter to invoke MODE depends upon the type of configuration operation you want to effect. There are four basic entry forms or methods of using MODE that support:

- configuring a parallel printer device,
- configuring a monochrome or color/graphics display device,
- configuring a serial printer or communications device, and
- remapping parallel output for use of a serial port.

The various command lines used for these four configuration functions are described in the paragraphs that follow.

Configuring a Parallel Device

To use MODE to configure your system for a parallel printer, enter a command line in the form:

```
MODE LPT#: [n] [, [m] [,P]]
```

where # is 1, 2, or 3 and designates the specific LPT device (LPT1:, LPT2:, or LPT3:) to be configured;
n specifies the number of characters per line (80 or 132);
m specifies the number of lines per inch (6 or 8); and
P invokes continuous retry on time-out errors.

If you do not specify values for the *n* and/or *m* parameters, the existing or default values are assumed. The power-up MODE default values for a parallel printer are 80 characters per line and 6 lines per inch.

The default handling for time-out errors is *no* continuous retry. If you wish to cause the system to make continuous retries in the event of time-out errors, then use the P parameter. Then, if desired during system operation, the retry loop can be stopped by pressing CTRL-BREAK. If you wish to turn off continuous retry for parallel printer operations after you have turned it on, you must enter another MODE command line that does *not* include the P parameter.

All parameters entered following the device name (LPT#:) must be separated from each other by commas (,). If you skip an optional parameter, you must enter a comma before the next parameter (*m* or P), if any, that you enter. Some examples follow.

Suppose you want to configure logical device LPT2: for the operation of a parallel printer at up to 132 characters per line, 6 lines per inch, and that you want the system to make continuous retries in the event of time-out errors. To configure LPT2: in this way, enter:

```
MODE LPT2:132,,P
```

at the system prompt and press **RETURN**. Later in the work session, if you want to turn off continuous retries for time-out errors without changing the other operational parameters, enter:

```
MODE LPT2:
```

at the system prompt and press **RETURN**. Remember that if you do not specify an *n* or *m* parameter, the preexisting (or MODE default) value will be assumed.

Configuring a Display Device

To use MODE to configure your system for a monochrome or color/graphics display device, enter a command line in the form:

```
MODE [n][,[m][,[T]][,s]]
```

where ***n*** is one of the following:

40	Sets the display width to 40 characters per line.
80	Sets the display width to 80 characters per line.
BW40	Sets the display mode to black and white (disables color burst) with 40 characters per line.
BW80	Sets the display mode to black and white (disables color burst) with 80 characters per line.
CO40	Sets the display mode to color (enables color burst) with 40 characters per line.
CO80	Sets the display mode to color (enables color burst) with 80 characters per line.
GR40	Sets the display mode to medium-resolution (320x200) graphics mode, with 40 characters per line.
GR80	Sets the display mode to high-resolution (640x200) graphics mode, with 80 characters per line.
MONO	Sets the display mode to monochrome. In this mode, the screen display width is 80 characters per line.

m is one of the following:

R	Shifts the display right.
L	Shifts the display left.

T causes MODE to produce a test pattern so that you can check display alignment before terminating MODE.

s specifies the scrolling mode, and is one of the following:

0	Software scroll mode.
1	Jump scroll mode.
2	Smooth (hardware) scroll mode.

When you use MODE in this way, select the *n* parameter value applicable for your peripheral and the operation(s) you wish to perform. The MONO display mode should be used only if you have a monochrome monitor. The display width parameters (40 and 80) may be used for either monochrome or color monitors. Other modes are applicable to color monitors and allow you to use the monitor in black and white, color, or color graphics (medium or high resolution) modes. Note that if you are working in the GR80 graphics mode and enter a MODE 40 command, the display will be changed to the GR40 mode. The reverse is also true. That is, if you are in the GR40 mode and enter a MODE 80 command line, the display will be changed to the GR80 mode. Similar results are obtained with the 40 and 80 parameters when the BW40, BW80, CO40, and CO80 modes are used.

If you wish to align the display produced by your device, enter an *m* parameter. For an 80 characters-per-line display, the *m* parameter shifts the display two character positions to the right (R) or left (L). For a 40 characters-per-line display, the *m* parameter shifts the display one character position to the right (R) or left (L).

If you wish to see a test pattern so that you can check (and adjust, if necessary) display alignment, enter the T parameter in conjunction with the *m* parameter. Before MODE terminates, a test pattern will be displayed and you will be prompted to indicate whether or not the display is aligned properly. If you enter **Y** in response to the prompt, MODE terminates. If you enter **N** at the prompt, then the shift (R or L) you specified will be repeated and the test pattern and prompt will be displayed again. Thus, you are given the opportunity to further shift the display in the same direction without having to reinvoke MODE.

NOTE: The T parameter can *not* be entered unless the *m* parameter is entered. The *m* parameter, however, may be entered with or without the T parameter.

Use the *s* parameter to specify the scrolling mode to be used. Enter 0, 1, or 2 to specify software, jump, or smooth (hardware) scroll mode, respectively. Software scroll mode is the default. This mode may be used in any display mode (as specified by the *n* parameter) and will be assumed if you do not enter a value for the *s* parameter. There are some restrictions on the use of the jump and smooth scroll modes, as shown in Table 11.2a.

Whenever you enter more than one parameter following the command name (MODE), the parameters must be separated by commas (,). Additionally, if you skip one or more optional parameters and then enter others, then you must enter a comma for each parameter that is omitted unless the omitted parameter is at the end of the command line.

Table 11.2a. Valid Scroll Modes

		SCROLL MODE		
		SOFTWARE	JUMP	SMOOTH
VIDEO MODE	40 BW40 CO40	VALID	NOT VALID	NOT VALID
	80 BW80 CO80	VALID	VALID	NOT VALID
	GR40	VALID	VALID	VALID
	GR80	VALID	VALID	VALID
	MONO	VALID	NOT VALID	NOT VALID

As an example, suppose you wish to configure your color monitor display for full color operation and 40 characters per line, and that you wish to align the display. To do this, enter:

```
MODE CO40,R,T
```

at the system prompt and press **RETURN**. The system is configured to support a 40-column color display and the display is shifted one character position to the right. Since no *s* (scroll) parameter was entered, the default value (0, software scroll mode) is assumed.

MODE displays a test pattern and prompts you to indicate whether the display alignment is correct. If it is, enter **Y** at the prompt. MODE will terminate and the system prompt will be displayed.

If the alignment is not correct (if the display must be shifted another character position to the right), enter **N** at the prompt. The display will be shifted another character position and then the test pattern and prompt will be displayed again. Note that if you wish to shift the display back to the left, you must enter another MODE command line that includes the L value for the *n* parameter.

NOTE: If you are working in text mode and use a MODE command (MODE *n*,L,T) to shift the display to the left, and if you subsequently enter graphics mode, the display may appear to be empty or nonfunctional. This may occur because a text screen can be wider than the effective area of a graphics mode display. (That is, the right margin of a text display may be out of the range of a graphics display.) If this occurs, simply enter the command **MODE R,T** and keep entering **N** in response to the test screen prompt until the display reappears in the proper alignment.

Configuring a Serial Device

To use MODE to configure your system for a serial printer or communications device, enter a command line in the form:

```
MODE COMn:baud[,[parity][,[databits][,[stopbits][,P]]]]
```

where ***n*** is either 1 or 2, and designates the specific COM device (COM1: or COM2:) to be configured;
baud is the speed (baud rate) at which your microcomputer will communicate with the device (110, 150, 300, 600, 1200, 2400, 4800, or 9600);
parity specifies whether odd (O), even (E), or no (N) parity is to be used;
databits specifies word length in bits, exclusive of any parity or stop bits, and may be 7 or 8;
stopbits specifies the number of stop bits (1 or 2) required; and
P specifies that you are configuring the COM*n*: device for a serial printer and causes the system to make continuous retries in the event of time-out errors.

When defining the protocol for a serial device, you must always specify the logical device (COM1: or COM2:) to be configured and the baud rate to be used. There are no default values for these parameters. Other parameters are optional; if you do not specify a value for any parameter(s) other than the device name and baud rate, then the MODE default value(s) will be assumed. The default values are:

- even parity (E);
- 7 data bits;
- 2 stop bits if the baud rate is 110; 1 stop bit if the baud rate is *not* 110; and
- no continuous retry on time-out errors.

Command line parameters entered following the device name must be separated from each other by commas (,). If you do not specify a value for an optional parameter (if you elect to use a default value), you must still enter a comma at that parameter's position in the command line.

When you specify the baud rate, you are not required to enter all digits in the baud rate. Only the first two characters of a valid baud rate (110, 150, 300, 600, 1200, 2400, 4800, or 9600) are required. You may enter the whole 3- or 4-digit number; however, any characters entered after the first two digits are ignored by MODE. Thus, if the desired baud rate is 110, you may enter 11 or 110; if the desired baud rate is 1200, you may enter 12, 120, or 1200, and so on.

If you wish to cause the system to make continuous retries in the event of time-out errors, specify the P parameter. Then, if desired during system operation, the retry loop can be stopped by pressing CTRL-BREAK. If you wish to turn off continuous retry for serial printer operations after you have turned it on, you must enter another MODE command line that does *not* include the P parameter.

As an example, suppose you wish to configure logical device COM2: for a serial device that operates at 1200 baud, even parity, 7 data bits, and 2 stop bits. Also suppose that you do not want continuous retry on time-out errors. To configure COM2: in this way, enter:

```
MODE COM2:1200,E,7,2
```

at the system prompt and press **RETURN**. As an alternative, you could enter the command line:

```
MODE COM2:12,,,2
```

In this case, only the significant digits in the baud rate are entered and the default values of even parity and 7 data bits are assumed.

If you wished to use default values to configure logical device COM1: for a serial printer operating at 300 baud and you wished to invoke continuous retry on time-out errors, you could enter:

```
MODE COM2:30,,,,P
```

at the system prompt and press **RETURN**. COM2: would then be configured for printer operation at 300 baud with even parity, 7 data bits, and 1 stop bit. If you wished to interrupt continuous retry on a printer time-out error, you could press CTRL-BREAK.

If you wished to turn off continuous retry without changing any other operational parameters, you would have to enter the command line:

```
MODE COM2:30
```

Remapping Parallel Output to a Serial Port

To use MODE to remap parallel output such that it is sent to a serial port, enter a command line in the form:

```
MODE LPT#:=COMn
```

where *#* is 1, 2, or 3 and designates the specific LPT device (LPT1:, LPT2:, or LPT3:) to be mapped to a serial port; and
n is either 1 or 2 and designates the specific COM device (COM1: or COM2:) to which the specified LPT device will be mapped.

This use of the MODE command causes all output directed to the LPT device to be redirected to the specified COM device. Before you can use MODE in this way, you must first configure your system for the COM device you will specify. If the actual COM device will be a printer, you must configure the logical COM device with a command line that includes the P parameter.

To disable remapping of an LPT device, you must enter a command line in the form **MODE LPT#:[*n*][,*m*]** as described in Configuring a Parallel Device in this section.

For example, suppose you wish to remap LPT1: such that all output that would normally be sent to that device will instead be sent to a serial printer connected to the COM2: port. Before you remap the parallel device, you must first configure the COM2: port by entering a MODE command in the form:

```
MODE COM2:baud[,[parity][,[databits][,[stopbits][,P]]]]
```

Use of this command entry form is described under Configuring a Serial Device in this section. Remember that if the actual serial device that will be used is a printer, you must enter the P parameter.

Once you have configured the COM2: device, you can remap the parallel device by entering:

```
MODE LPT1:=COM2
```

at the system prompt and pressing **RETURN**. In subsequent system operation, all output that previously would have been sent to LPT1: will be sent to COM2:. If you wish to restore normal system use of LPT1:, you must enter a command in the form:

```
MODE LPT1:[n][,m]
```

where ***n*** is the desired number of characters per line, and
m is the number of lines per inch vertical spacing.

This command entry form is described under Configuring a Parallel Device in this section. After you enter this command, the LPT device output will no longer be remapped to the serial device. However, the serial device as you configured it will still be available for use by the system.

Error Messages

```
Illegal device name
```

EXPLANATION: This message is displayed if you entered an invalid LPT or COM device name as part of the MODE command line. Reenter the command line, making sure that you use valid MS-DOS device names that exist in your hardware system configuration.

```
Invalid baud rate specified
```

EXPLANATION: This message is displayed if you entered an invalid baud rate value in the MODE command line. Reenter the MODE command line, making sure you specify an allowable baud rate.

```
Invalid parameters
```

EXPLANATION: This message may be displayed if you made a syntax error in the MODE command line you entered, or if you entered too few or too many command line parameters. Reenter a valid MODE command line.

MORE (Transient)

Purpose

Sends output to the terminal one screen at a time.

Entry Form

```
MORE
command | MORE
```

where ***command*** is the command or function whose output you want to pipe to MORE; and

| is the pipe that causes the output of *command* to be input to MORE.

Preliminary Concepts

This command is a filter that reads from standard input (such as a command input at the keyboard) and displays one screen of information at a time. When the screen is filled, the display of information pauses and the last line on the screen displays

```
-- More --
```

To display the next screen of information, press **RETURN**. This process continues until all the input data to MORE has been read.

Command Line Entry

As a filter, MORE is generally used following another command and a pipe to pace command output so that it can be easily read by the user. Thus, MORE usually is used in a command line having the form

```
command | MORE
```

where *command* is the operation whose output you wish to filter through MORE. (For more information on pipes and filters, refer to Chapter 8, "Input/Output Features.")

Using MORE to Display Files

Because the MORE command enables you to view one screen of information at a time at your own pace, it is useful for reviewing lengthy files. For example, suppose you have a data file MYDATA.DAT on the disk in drive D and wish to review the file's contents without having to use a text editor or print the entire file. Assuming that MORE is on the disk in the default drive, you could easily review the file by entering

```
MORE <D:MYDATA.DAT
```

or

```
TYPE D:MYDATA.DAT | MORE
```

and pressing **RETURN**.

In the first example (which is the most efficient way to use MORE in an instance such as this), the left-angle bracket (<) causes MORE to read file D:MYDATA.DAT as input. (If no input/output redirection or pipes are used, MORE reads the standard input.) The file is filtered through MORE and its contents displayed one screen at a time.

In the second example, the command line pipe (|) directs the output of the TYPE operation to MORE. This causes the file to be displayed one screen at a time rather than continuously, as it would be if TYPE were used alone.

In either case, the display pauses when the screen is filled and the last line of the screen displays

```
-- More --
```

To see the next screen of information, press **RETURN**. Each time `--More --` is displayed, you may continue by pressing RETURN. When the entire file has been displayed, the system prompt is displayed again.

If you wish, you may use the SHIFT-PRTSC function to selectively print the screens of information displayed by using MORE. (For more information about printing screen displays, refer to Chapter 8, "Input/Output Features.")

Error Message

```
Invalid DOS version
```

EXPLANATION: This message is displayed if you try to use the MORE command with an incompatible version of MS-DOS. MORE will execute only under MS-DOS version 2 or higher.

PATH (Resident)

Purpose

The PATH command is used to specify directories, other than the current one, to be searched in the event that a transient command cannot be found in the current working directory.

Entry Form

```
PATH [d:][pathname [;[d:]pathname]...]
```

where ***d*** is the letter of the designated disk drive; and ***pathname*** is a sequence of characters of the form:

`[\] [directory] [\directory...]`

Optionally you may use the MS-DOS shorthand notation shown below in lieu of *directory*:

.   MS-DOS uses this shorthand symbol to indicate the name of the current (working) directory in all hierarchical directory listings. MS-DOS automatically creates this entry when a directory is made.

..   MS-DOS uses this shorthand symbol to indicate the name of the current directory's parent directory. MS-DOS automatically creates this entry when a directory is made.

NOTE: The two shorthand symbols do not exist in the root.

Preliminary Concepts

You can only access a transient command if that command is present in the current working directory or by specifying the PATH command. In order to access the system's transient commands from your current working directory, MS-DOS provides the PATH command. The PATH command allows you to specify alternate paths to search for the transient commands from your current working directory to the the directory in which the transient commands are located. This is necessary if you want to access any of the MS-DOS transient commands from your current working directory.

The PATH command will always search the current working directory first, then any directories specified in the PATH command. The PATH command will search the directories in the command line, beginning with the first path name specified. You may specify as many alternate path names for this command to search as you wish, up to a total line length of 128 characters.

The default setting of the PATH command at bootup is "No Path." This means that only the current working directory is searched for transient commands.

Command Line Entry

If you are in the current working directory and all of the MS-DOS transient commands are in another directory, you must tell the operating system to choose the correct path to that directory.

For example, if you are in directory \BIN\USER\JOE, the transient commands are in directory \BIN, and you wish to set a path to search for transient commands, enter

```
PATH \BIN
```

and press **RETURN**. MS-DOS will search the \BIN directory whenever you execute any of the transient commands.

NOTE: You only have to specify PATH one time during your session at the microcomputer. The PATH command that you specify will remain in effect until you specify another, or until you reset MS-DOS.

If you need to know what the current setting of PATH is, enter

```
PATH
```

and press **RETURN**. If your current path is \BIN, the screen would display

```
A>PATH
PATH=\BIN
```

You can instruct the PATH command to search more than one path by specifying several path names separated by semicolons. For example, entering

```
PATH \BIN\USER\JOE;\BIN\USER\SUE;\BIN\DEV
```

and pressing **RETURN** tells MS-DOS to search the directories specified by the above path names to find transient commands. The PATH command searches the path names in the order specified in the command line.

If you do not specify a path, MS-DOS will set the *nul path*, meaning that only the current working directory will be searched for transient commands.

You do not need to specify a path if you are in the directory that contains the transient commands.

Advanced Concepts

When you are working with more than one directory, it is convenient to place all of the MS-DOS transient commands in a separate directory, so they can be easily accessed from any directory within the system.

If you have all the transient commands in their own directory, \BIN for example, it is quite easy to set that path at the beginning of any session.

Error Messages

```
Out of environment space
```

The "environment" string space has exceeded its allocated space. Use the SET command to delete unnecessary strings in the "environment."

PAUSE (Resident, Batch-Processing)

Purpose

Suspends execution until any key except CTRL-BREAK is pressed. If the command is used in a batch file and CTRL-BREAK is pressed while PAUSE is in effect, batch file execution is aborted.

Entry Form

```
PAUSE [remark]
```

where ***remark*** is an optional user-defined remark or comment.

Preliminary Concepts

The resident, batch-processing commands are most often executed from within a batch file, although they may be used directly from a command line at the system prompt in some instances. Refer to the section Automatic Command Entry in Chapter 5, "Command Features," for a complete explanation of batch file creation and execution.

NOTE: You must always end each line within a batch file by pressing the RETURN key.

Command Line Entry

This command is invoked simply by entering the command name, PAUSE. Optionally, a *remark* may be included on the same line as the PAUSE command. The remark will be displayed on the screen verbatim when the command is encountered during batch file execution. This optional parameter enables you to create an informational message or prompt for batch file users. (Refer to Chapter 5, "Command Features," for information on creating and using batch files.)

Using the PAUSE Command

During the execution of a batch file, you may need to change disks or perform some other function requiring that the program temporarily stop. The PAUSE command may be included in any batch file to accomplish this.

When the system encounters a PAUSE command, execution stops and the screen displays

```
Strike a key when ready . . .
```

Pressing any key except CTRL-BREAK causes the system to resume execution. If you press CTRL-BREAK and the PAUSE command was in a batch file, the screen displays

```
Terminate batch job (Y/N)?
```

If you wish to terminate the batch job, enter **Y**. Execution of the remainder of the batch file is aborted and control returns to the operating system command level. In this way, the PAUSE command can be used to divide a batch file into parts and give the user the option of ending execution at an intermediate point.

If you do not wish to terminate the batch job, enter **N**. Execution of the batch file will resume.

When you create your own batch files, you will find that the PAUSE command is indispensable for some procedures. PAUSE can allow you to accomplish procedures in one batch file that otherwise might require more than one batch file. For example, many procedures require some type of break to allow the user to change disks, look up information, or turn switches on/off between certain steps.

Creating a Message or Prompt

A remark or comment may be entered on the same line as the PAUSE command; the *remark* will be displayed on the screen when the PAUSE command is encountered. This enables you to create an informational message or prompt for batch file users. For example, suppose you want to use the PAUSE command to suspend execution of the batch file so you can change disks at a given point. You could enter **PAUSE Change data disk...** in the batch file at the appropriate point. Then, when the command is encountered during execution of the batch file, the screen displays

```
Pause Change data disk...
Strike a key when ready . . .
```

You could then change the disk and press any key except CTRL-BREAK to continue batch file execution. If you wished to terminate batch file execution, you could press **CTRL-BREAK**. The screen would then display

```
Terminate batch job (Y/N)?
```

To end batch file execution, you would enter **Y**. If you decided not to terminate batch file execution, you could enter **N** at this prompt and execution would resume.

Any remark entered as part of the PAUSE command line is displayed before the system `Strike a key when ready . . .` prompt.

PRINT (Transient)

Purpose

Invokes spool or background printing to print specified text files while other MS-DOS commands are being processed.

To use the PRINT command, you must have a device connected to your system and your system must be properly configured to interface with the device. (For information on CONFIGUR, refer to the appropriate section of this chapter.)

Entry Forms

```
PRINT
PRINT filespec[/x] [filespec[/x]]...
```

where ***filespec*** is the file specification for a file or group of files that you wish to print; and

/x is one or more of the following switches:

/A	Abort print (cancels the associated file and all following file specifications (until an /S switch is encountered) from the print queue).
/C*n*	Copies (causes PRINT to produce *n* copies of the associated file; *n* may be any value from 1 to 255, inclusive).
/F	Form feed (causes PRINT to issue a form feed at the end of each copy of the associated file).
/L*n*	Left margin (sets the left margin for the associated file at the *n*th column).
/P*n*	Page length (sets the page length for the associated file at *n* lines).
/R*n*	Right margin (sets the right margin for the associated file at the *n*th column).
/S	Spool print on (adds the associated file and all following file specifications (until an /A switch is encountered) to the print queue).
/T	Terminate (terminates the PRINT function by eliminating all files from the print queue).

Preliminary Concepts

The PRINT command enables you to place up to 30 files in a print queue for spool or background printing (that is, the files will be printed on a time-available basis as the system is executing other commands). Once the command has been invoked and there are files in the print queue, you may alter the contents of the queue without having to enter the entire command sequence again for the files already in the queue.

Command Line Entry

The parameters of the PRINT command line are described below. As a minimum, you must always enter the command name. If you enter *only* the command name, PRINT, the screen will display the names of the files currently in the print queue, if any. If no files are in the print queue, the screen will display

```
PRINT queue is empty
```

File to Print

When you wish to print a given text file, you must enter the *filespec* for the file following the command name. The *filespec* and the command name (PRINT) should be separated by a space. If the file to be printed is in the current directory of the default disk, enter the primary file name and extension, if one exists. If the file is on a disk other than the default disk, you must include the appropriate drive name (*d*) as part of the file specification. A path name may also be used for *filespec* if the file exists in a directory other than the current directory.

You may specify up to 30 files for inclusion in the print queue with the PRINT command. If you specify more than one file, each *filespec* is separated from the other(s) by a space. Switches regarding the printing of a given file must follow immediately after **that** file; switches may *not* be placed at the end of the command line for all files in a series. If you put all switches at the end of the command line when a series of files is specified, they will affect only the printing of the last file in the series.

Wildcard characters may be used in file specifications entered as part of the PRINT command line. If you use wildcard characters in a file specification and switches are entered following the *filespec*, the switch functions are invoked for every file printed that matches the *filespec*.

Switches

The switches described below are provided to enable you to control the PRINT function, and to enable you to format the text column width and page length for the file(s) you print. Some switches affect the printing of only one file. Others (such as the /A and /S switches) may affect a series of files.

Switches must be entered on the command line immediately after the (first) file specification to which you want the switch function to apply. No delimiter other than the switch character (/) that is entered as part of each switch should be used between the file specification and associated switch(es). If more than one file specification is entered on a command line, each *filespec* and its associated switches are separated from the adjacent ones by a space.

/A—Abort Print

This switch is used to cancel (remove) a specified file or files from the print queue. When you enter an /A switch, the associated file and all following files (if any) up to the end of the command line or until an /S switch is encountered will be removed from the print queue.

/C*n*—Copies

Normally, PRINT only prints one copy of each file in the print queue. The /C*n* switch, where *n* is an integer value from the range of 1 to 255, inclusive, enables you to specify how many copies of a given file will be printed. Thus, if you wanted three printed copies of a given file, you would enter the switch /C3 immediately following the *filespec* parameter for that file.

/F—Form Feed

This switch causes PRINT to issue a form feed at the end of each copy of the associated file. Thus, when you enter /F for a requested file, PRINT will cause the printer to move the paper to the top of a form at the end of the file, before printing the next copy (if multiple copies were requested with the /C*n* switch) or file.

/L*n*—Left Margin

Normally, PRINT will begin printing each line of the requested file at the first column for which the printer is set. With the /L*n* switch, where *n* is an integer value, you may set the left margin of the file at any column allowed by your printer that you wish. When you enter /L*n* for a requested file, PRINT will begin each line of that file *n* columns to the right of the first column.

This switch only affects the printing of its associated *filespec*.

/P*n*—Page Length

Normally, PRINT will not check for page length when printing requested files; printing proceeds continuously across pages or forms. With the /P*n* switch, where *n* is an integer value, you may specify the number of lines to be contained on each printed page. When the number of lines specified by *n* have been printed on a page, PRINT will issue a form feed and begin printing on the next page or form.

NOTE: If the associated file contains embedded form feeds, the /P*n* switch will *not* override them.

/R*n*—Right Margin

Normally, PRINT will print lines up to the maximum width (default line width) allowed by the printer you use. (If the text file you print contains RETURN's at the end of each line, PRINT will begin a new line each time it encounters a RETURN.) With the /R*n* switch, where *n* is an integer value, you may set the right margin that will be used when the requested file is printed. When you enter /R*n* for a requested file, PRINT will end each line of the file at the *n*th column and continue the text on the next line.

NOTE: If the associated file contains embedded RETURN's, the /R*n* switch will *not* override them.

/S—Spool Print On

This switch is used to turn on the PRINT function, such that the associated file and all following files will be added to the print queue. Files are printed in the order in which they are specified. Printing will continue until the queue is empty, until you enter an /A (Abort) switch, or until you enter a /T (Terminate) switch, whichever occurs first.

This switch is normally used when you have already invoked the PRINT command for a file or files, and wish to add more files to the print queue. It is not required if you invoke the PRINT command when the print queue is empty, when no files are being printed. (Refer to Printing a Series of Files and to Manipulating Files in the Print Queue for examples of /S switch use.)

/T—Terminate

This switch flushes (deletes all files from) the print queue, terminating the PRINT function.

Printing a Series of Files

Suppose you have a series of text files on the disk in drive B that you wish to print while you continue with other MS-DOS functions. If the files are already formatted with embedded RETURN's and form feeds for line and page lengths, respectively, you may invoke the PRINT function for the files using a minimum of switches. If you wish to print two copies of LETTER.CTR, and one copy each of MYFILE, WOM.BAT, and REPORT.DOC, enter

```
PRINT B:LETTER.CTR/C2/F B:MYFILE/F B:WOM.BAT/F B:REPORT.DOC/F
```

and press **RETURN**. If this is the first time you have run PRINT during the current work session, the screen will display

```
Name of list device [PRN]:
```

At this prompt, you may specify as the PRINT output device any current device. The default is PRN. To specify the default, press **RETURN**. If you wish to specify another device, enter the device name and press **RETURN**. (Refer to Chapter 8, "Input/Output Features," for information on device names.) After a device is specified, the PRINT function begins and the screen displays

```
Resident part of PRINT installed

        B:LETTER  .CTR is currently being printed
        B:MYFILE  .    is in queue
        B:WOM     .BAT is in queue
        B:REPORT  .DOC is in queue
```

This display changes to reflect the status of files in the queue. When printing of a file is complete, that file is removed from the top of the list and the next file in the queue is printed. In this example, the /F switch following each file specification in the command line causes PRINT to issue a form feed at the end of each file. The /C2 switch following B:LETTER.CTR causes PRINT to print this file twice before going to the next file (B:MYFILE) in the queue.

Suppose that, during execution of the above PRINT function, you discovered that you left two files out of the print queue that you do want to print. You could add the files to the queue by using the /S switch. That is, by entering

```
PRINT B:FORGOT1.DOC/S B:FORGOT2.DOC
```

at the system prompt and pressing **RETURN**. The files will be added to the print queue and will be displayed on the screen with other files in the queue.

Suppose that, during execution of the above example PRINT function, you wished to delete the file WOM.BAT from the print queue. You could accomplish this by using the /A switch, by entering

```
PRINT B:WOM.BAT/A
```

at the system prompt and pressing **RETURN**. This causes PRINT to delete the file from the print queue. The status of other files in the print queue in not affected. If the file is already being printed when you enter the command, printing of the file is aborted.

When you abort a file by using the /A switch, the printer will print

```
filespec Cancelled by operator
```

where *filespec* is the file specification of the aborted file.

Displaying the Contents of the Print Queue

Any time you wish to review the contents of the print queue, you may do so by entering

```
PRINT
```

and pressing **RETURN**. If there are files in the queue, the screen will display a list of the files in the following form:

```
d:filename.ext is in queue
d:filename.ext is in queue
       .
       .
       .
d:filename.ext is in queue
```

The list is followed by the system prompt.

If there are no files in the queue, the screen displays

```
PRINT queue is empty
```

and then the system prompt is displayed again.

Manipulating Files in the Print Queue

Once the PRINT function has been invoked for a file or files, you may use additional command line entries and switches to add and subtract files from the print queue. For example, suppose you have a series of files named TEMP1.TST, TEMP2.TST, TEMP3.TST, TEMP4.TST, and TEMP5.TST in the print queue, and that the files are all on the disk in the default drive. If you wanted to remove the first three files from the print queue, you could enter

```
PRINT TEMP1.TST/A TEMP2.TST TEMP3.TST
```

and press **RETURN**. The files would be removed from the queue, and files TEMP4.TST and TEMP5.TST would remain in the queue for printing.

If only files TEMP4.TST and TEMP5.TST were in the print queue, you could delete them and add the other three files to the queue by entering

```
PRINT TEMP4.TST/A TEMP5.TST TEMP1.TST/S TEMP2.TST TEMP3.TST
```

and pressing **RETURN**.

To clear the print queue of all files and end the PRINT function, you would enter

```
PRINT /T
```

and press **RETURN**. Also note that you could empty the print queue and put more files in the queue with one command, such as

```
PRINT /T *.ASM
```

This command would clear the print queue, terminating the current PRINT function, and then add all files on the default disk with the extension .ASM to the queue (up to the print queue maximum of 30 files).

Error Messages

The messages that you may see displayed on your screen or printed while using the PRINT function are described below. Messages listed are displayed on the screen unless otherwise specified.

```
All files cancelled by operator
```

EXPLANATION: This message is printed if you enter a /T switch during execution of the print function.

```
d:filename.ext Cancelled by operator
```

EXPLANATION: This message is printed if you delete a file (*d:filename.ext*) from the print queue by using the /A switch.

```
Drive not ready
```

EXPLANATION: This message is displayed if an error occurs when PRINT is trying to access a disk. When this happens, PRINT will keep trying to access the disk unless another error also occurs. Any other error causes printing of the current file to be cancelled, and the message `filespec Cancelled` to be printed.

```
filespec File not found
```

EXPLANATION: This message is displayed if a specified file (*filespec*) is not found by the system for input to the PRINT function.

`Invalid value for /x ignored`

EXPLANATION: This message is displayed if you entered an invalid or unacceptable value for one of the switches requiring a numeric parameter, (that is, the /C*n*, /L*n*, or /R*n* switch). When this occurs, the switch will be ignored and the PRINT function will proceed as though the switch had not been entered.

`List output not assigned to a device`

EXPLANATION: This message is displayed if you enter an invalid device name at the `Name of list device [PRN]:` prompt when the PRINT command is first invoked. When this occurs, the system prompt is displayed again and you must reinvoke the PRINT command. (Refer to Chapter 8, "Input/Output Features," for more information on valid device names.)

`Name of list device [PRN]:`

EXPLANATION: This prompt is displayed the first time you invoke the PRINT command during a given work session. You must specify a valid device name to which PRINT output will be sent. The default device is PRN; it may be specified simply by pressing RETURN. If you wish to specify an alternate device, enter the three-letter device name and press RETURN. (For more information on valid device names, refer to Chapter 8, "Input/Output Features.") When a valid device has been specified, the screen displays

`Resident part of PRINT installed`

and execution of the PRINT function begins.

`No files match filespec`

EXPLANATION: This message is displayed when a file specification has been entered for a file to add to the print queue, but the system cannot find a file to match the specification.

NOTE: If you specify a file to be deleted from the print queue but there are no files matching that specification in the print queue, no error message will be displayed.

```
PRINT queue is empty
```

EXPLANATION: This message is displayed if you request a display of print queue contents and there are none, or when you clear the queue and terminate the PRINT function by entering a /T switch.

```
PRINT queue is full
```

EXPLANATION: This message is displayed when you attempt to place more than the allowable maximum of 30 files in the print queue. The first 30 files specified will be accepted, but any files beyond that number will have to be added to the queue at a later time with another PRINT command or commands.

PROMPT (Resident)

Purpose

The PROMPT command is used to change the MS-DOS system prompt.

Entry Form

```
PROMPT[$][prompttext]
```

where ***prompttext*** is the new prompt you specify to replace the current system prompt. The new prompt is created by specifying some combination of the following:

a dollar sign followed by the special prompt characters found in Table 11.3; and

any text you also want to appear in the new prompt.

Preliminary Concepts

This command allows you to change the MS-DOS system prompt. If no *prompttext* is entered, the prompt will be set to the default system prompt, which is the default drive name followed by the "greater than" sign (A>). You can set the prompt to a special prompt by using the characters illustrated in Table 11.3.

NOTE: The default system prompt is displayed every time you boot up. Any special prompt created with the PROMPT command is *not* recorded on disk.

Command Line Entry

The following table lists the characters you can use to produce a customized system prompt. They must all be preceded by a dollar sign ($) within the PROMPT command.

Table 11.3. Valid Characters Used in the PROMPT Command

SPECIFY THIS CHARACTER	TO GET THIS PROMPT
$	The $ character
t	The *current time*
d	The *current date*
p	The *name of the current directory of the default drive*
v	The *MS-DOS version number*
n	The *letter of the default drive*
g	The > character
l	The < character
b	The | character
_	A *CR LF sequence* (This instructs the prompt to go to the beginning of a new line on your screen.)
s	A *space* (leading only)
h	A *backspace* (erases the previous character)
e	An *escape* (ASCII code X1B)

The above characters (which may be entered by using upper- or lowercase alpha characters) when preceded by a dollar sign

PROMPT

($), will cause the listed items to be displayed. All other characters entered will appear within the new prompt just as you typed them.

The length of the entire command line is 128 characters. Subtracting seven (six for PROMPT and one for the space which follows) leaves 121 as the maximum characters in any prompt line.

The following examples illustrate the use of the PROMPT command.

Entering the command

```
PROMPT $n$g
```

and pressing the **RETURN** key, will set the normal MS-DOS prompt.

x>

where *x* is the letter of the current drive.

If you wish to have the current time and date displayed as the prompt, you would enter the command

```
PROMPT $_Time = $t$_Date = $d
```

and press the **RETURN** key. This command sets a two-line prompt which displays:

```
Time = (current time)
Date = (current date)
```

To return to the default system prompt from any special prompt you have created, type

```
PROMPT
```

and press the **RETURN** key. This will return the default system prompt that appeared at boot up.

Advanced Concepts

If you want to create a prompt that begins with any of the MS-DOS command line delimiters, such as semicolon, blank, or others; you can precede it with the $ character. When this is done, the specified symbol following the $ character will be treated as the first character in the new prompt.

You may also wish to create a new prompt that automatically displays at boot up. This can be done by including the PROMPT command in the file AUTOEXEC.BAT. Refer to Chapter 5, "Command Features," for a complete discussion of this procedure.

PSC (Transient)

Purpose

The various PSC (print screen) commands are used to output all graphics and special characters from the screen to the printer.

Entry Form

```
PSCprintername
```

where ***printername*** is one of the following printer options:

IDS —for use with the IDS Prism Printer
MPI —for use with the MPI Printer
MX80 —for use with the Epson MX-80 with the graphics option
OKI —for use with the Okidata printer
P920 —for use with the Printek 920
TS315 —for use with the Transtar 315 Color Printer

Preliminary Concepts

The PSC utility enables you to print anything, including graphics or special characters, displayed on your screen. The PSC utility will work with any of the printers listed above.

In order for PSC to output the information to the PRN (LPT1) device (usually a printer), you must have properly configured the PRN device with the CONFIGUR program.

NOTE: Your microcomputer must be in the *graphics mode* before any of the PSC utilities will print graphics. Most application programs that use graphics will already be in graphics mode, and you may put the microcomputer into the graphics mode from BASIC by issuing the SCREEN statement.

The different versions of PSC are specific to the printers shown. Therefore, you must specify PSCMX80, for example, if you wish to dump a screen display to an MX80 printer.

The PSC utility gives you the option of printing all information shown on your screen, instead of just the printable ASCII characters available with the default SHIFT-PRTSC key sequence and the CTRL-PRTSC key sequence.

Because the various versions of PSC print *all* information appearing on the screen, they tend to operate more slowly than the default SHIFT-PRTSC key sequence.

Command Line Entry

The PSC utilities must first be loaded into memory before they can be used. They are loaded into memory by entering

```
PSCprintername
```

and pressing **RETURN**

where ***printername*** is one of the printer options. PSC need only be executed once. After it is executed, it remains in memory until the system is turned off or until MS-DOS is again booted.

Once PSC has been executed (loaded into memory), it can be called to print full screen displays at any time. This is accomplished by simultaneously pressing the SHIFT and PRTSC keys. This causes all characters appearing on your screen to be output to the PRN device. Remember, you must have properly configured the PRN (printer) device. This is particularly important, as your printer will "lock up" if it is not properly configured or if you try to use any PSC parameter except the one specific to your printer.

RDCPM (Transient)

Purpose

The RDCPM (Read CP/M) command is used to read CP/M formatted disks and copy their files to MS-DOS formatted disks.

Entry Forms

```
RDCPM
RDCPM ?
RDCPM DIR d: [filename.ext] [/Z]
RDCPM [s:]sorcfile.ext [d:] |destfile.ext] [/Z]
```

where **?** causes the RDCPM help screen to be displayed;

DIR is used to read a CP/M formatted disk's directory;

d is the letter of the specified drive you want to read.

filename.ext is the optionally specified file name and extension to be read, from the CP/M formatted disk (this may be matched to an ambiguous file name by using the wildcard characters);

[*s*:]*sorcfile.ext* is the source drive and file name of the CP/M formatted disk you wish to copy;

[*d*:][*destfile.ext*] is the destination drive and file name to which you are copying; and
/Z is an optionally specified switch for reading/copying Zenith Data Systems CP/M formatted disks (the default setting is for IBM-PC CP/M-86 formatted disks).

Preliminary Concepts

RDCPM has two primary functions: to read a CP/M formatted disk's directory and to copy CP/M formatted files into the MS-DOS format.

RDCPM expands the usefulness of MS-DOS by allowing you to use the features of MS-DOS while taking advantage of documents and files created with CP/M.

NOTE: The RDCPM utility will read ONLY floppy disks containing CP/M files. It will NOT read CP/M files from a Winchester hard disk. However, RDCPM will copy to either a Winchester disk partition or a floppy disk.

The RDCPM command is set up to read IBM-PC CP/M-86 formatted 5.25-inch, 48-tpi, double-sided or single-sided disks. If you specify the /Z switch, RDCPM will read Zenith Data Systems CP/M formatted 5.25-inch, 48-tpi, double-sided or single-sided disks.

NOTE: The RDCPM utility will *not* read 96-tpi disks.

Command Line Entry

The RDCPM utility can be entered in three fundamental ways:

1. To read the directory of a CP/M formatted disk.

2. To copy a specified file from a CP/M formatted disk to an MS-DOS formatted disk, optionally using wildcard characters to specify an ambiguous file name to be matched.

3. To copy a specified file, or files from a CP/M formatted disk and optionally renaming the files in the process.

Examples

RDCPM may be invoked without specifying any parameters. If you type

```
A>RDCPM
```

and press **RETURN**. RDCPM returns the message:

```
                RDCPM version 2.0

Usage:  RDCPM source [destination]
-or-    RDCPM DIR drive
```

If you would like more detailed instructions about RDCPM's function, enter

```
A>RDCPM ?
```

and press **RETURN**. The following description of RDCPM will appear on your screen:

```
                RDCPM version 2.0

     RDCPM reads a file, file group, or directory from
a disk that has been formatted using a CP/M compatible
system and will copy the file(s) to a disk having been
formatted with MS-DOS.

     The RDCPM command is similar to the COPY command,
except that the source file(s) are expected to be on a
CP/M disk.
```

RDCPM

```
Usage: RDCPM source [destination]
 -or-  RDCPM DIR drive

Where "source" is a CP/M filespec, and "destination" is
an MS-DOS filespec or drive name.
```

To read the directory of a Zenith Data Systems CP/M formatted disk in drive B, enter

```
A>RDCPM/Z DIRB:
```

and press **RETURN**. This will give you the directory of the CP/M formatted disk in drive B.

To read a file, or group of files from a CP/M disk, you would follow the example below.

In this example we will display a directory of the .BAK files on our CP/M formatted disk, located in drive B. To begin, enter

```
A>RDCPM DIR B:*.BAK
```

and press **RETURN**. This tells RDCPM to read the directory of the disk in drive B and match the wildcard character (*) to all .BAK files on that disk.

The file name specification (where we specified *.BAK) may be any valid CP/M file name and extension (or wild cards) that represent the file(s) that you want to see in the directory. The RDCPM banner will display on the screen. Below that, the directory of the CP/M formatted disk will appear in CP/M directory form (with file names and extensions in four columns) on your screen.

To copy a CP/M formatted file or several files (wild cards are acceptable), enter

```
A>RDCPM [s:]sorcfile.ext [d:]destfile.ext
```

where [*s*:] is the optional source drive name;
sorcfile.ext is the CP/M source file name;

[***d***:] is the optional drive name for the MS-DOS formatted disk; and
[***destfile.ext***] is the optional new name for the file that is being copied to the MS-DOS destination.

NOTE: RDCPM used in this manner (to copy a CP/M file from a source disk that has been formatted with CP/M, to a destination disk that has been formatted with MS-DOS) operates in a similar manner to the MS-DOS COPY command. It does require that the source is a CP/M formatted floppy disk and that the destination is an MS-DOS formatted floppy disk or Winchester partition. You may also rename the copied files with this procedure, much as you would with the COPY command.

During the copy operation, RDCPM will display

```
RDCPM version 2.0
```

directly above the list of file names that are copied. This list is presented similar to the way COPY displays the file names that it is copying.

NOTE: The RDCPM utility will copy these CP/M formats:

Zenith Data Systems formatted

CP/M-85 SS/DD 48-tpi 5.25″ disks
CP/M-85 DS/DD 48-tpi 5.25″ disks
CP/M-86 SS/DD 48-tpi 5.25″ disks
CP/M-86 DS/DD 48-tpi 5.25″ disks

IBM formatted

CP/M-86 SS/DD 48-tpi 5.25″ disks
CP/M-86 DS/DD 48-tpi 5.25″ disks

Where SS = single-sided;
DS = double-sided; and
DD = double-density.

Error Message

```
Drive not available for CP/M reading
```

EXPLANATION: This message may appear if you try to use an invalid drive designation or if you try to use RDCPM to copy CP/M files from a Winchester hard disk.

RECOVER (Transient)

Purpose

Recovers a single specified file or all files on a disk containing bad sectors.

Entry Forms

```
RECOVER d:
RECOVER filespec
```

where ***d*** is the drive name identifying a drive in which there is a disk that you want to recover; and
filespec is the file specification of a single file that you want to recover.

Preliminary Concepts

The RECOVER command enables you to salvage the usable portions of a file or of all files on a disk that contains bad sectors. *Bad sectors* are media imperfections that can cause hard errors during disk access operations. *Hard errors* are conditions under which an operation fails after a number of repeated attempts. System recovery from a hard error usually brings an abrupt end to the operation that was being attempted.

The media imperfections that the system identifies as a bad sector may be damage to the disk itself or damage to the data stored on the disk, as may happen when a disk is inadvertently exposed to an electromagnetic field.

When you are unable to use a file or a disk because of bad sectors, you should run RECOVER for the file or for the disk. Generally, it is more efficient to recover one file at a time (as described under Recovering a Single File), than to recover an entire disk. This is because RECOVER does not distinguish between damaged and undamaged files when an entire disk is recovered, and you may have to access and rename files after recovery that were not damaged originally. Recovering an entire disk also removes the hierarchical directory structure if the disk contains subdirectories. If, however, an *entire* disk is damaged or if the disk directory has been damaged, you will probably want to run RECOVER for the disk rather than for selected files.

RECOVER will not recreate the data in damaged sectors of a file or disk, but will salvage as much as possible of the file or disk so that you may reedit or correct it for continued use. RECOVER prevents your having to recreate the entire file (or files) manually.

Command Line Entry

The parameters of the RECOVER command line are described below. You must enter one of the two described parameters in any given command line to recover either a specific file or an entire disk.

Drive Name

If a disk contains many bad sectors or if the directory has been damaged, you may recover the entire disk by specifying the drive name (*d*) of the drive in which the disk is located. Thus, to recover all files on a selected disk, enter

```
RECOVER d:
```

at the system prompt and press **RETURN**. *d* is the drive in which the disk is located. Recovery of an entire disk may be a lengthy process.

When you press RETURN, RECOVER begins reading the entire disk sector by sector, and writes the files it finds to the root directory, regardless of the directory in which they originated. Recovered files are sequentially assigned file names in the form FILE*nnnn*.REC, where *nnnn* is a four-digit number. The first file recovered has the file name FILE0001.REC, the second file recovered has the file name FILE0002.REC, and so on.

If there is not enough room in the root directory for all the files on the disk, RECOVER will display a message to that effect and store information about the unrecovered files in the disk's file allocation table. When there is more room in the root directory (such as after you have copied the recovered files to another disk and deleted them from the original disk), you can execute RECOVER again to regain the remaining files.

File Specification

If there is one file containing a bad sector and you wish to recover as much information from the file as possible, you may enter at the system prompt a command line in the form

```
RECOVER filespec
```

where ***filespec*** is the file specification of the file you want to recover.

If the file is in the current directory of the default disk, enter the primary file name and extension, if one exists. If the file is in another directory and/or on another disk, you must enter the full file specification, including the path name and/or appropriate drive name.

Do *not* use wildcard characters in the file specification for the file to be recovered. When the RECOVER command is entered with the *filespec* parameter, only one file will be recovered. If you enter a file specification that includes wildcard characters, only the first file found that matches the specification will be recovered.

When you press **RETURN**, the file is read sector by sector and rewritten to the disk and directory in which it originated. When MS-DOS encounters a bad sector, that sector is skipped and the cluster containing the bad sector is marked so that MS-DOS will not allocate data to that sector either during the current recovery function or in future use of the disk until it is formatted again.

Recovering a Single File

Suppose you have a lengthy text file on a disk and the disk has developed bad sectors that prevent you from easily accessing or using the file. If the file is on the disk in drive B, you could recover it by entering

```
RECOVER B:DAMAGED.TXT
```

and pressing **RETURN**. The screen displays

```
Press any key to begin recovery of the
file(s) on drive B
```

If you want to abort the RECOVER operation at this point, you may do so by pressing **CTRL-BREAK**. The system prompt will be displayed.

If you want to continue with the RECOVER operation, press any alphanumeric key (that is, any nonfunction or noncontrol key) and the recovery operation will begin. When the file has been recovered, the screen displays

```
nnnn of NNNN bytes recovered
```

followed by the system prompt. In this message, *nnnn* is the number of bytes contained in the file that the system created in the recovery operation; *NNNN* is the number of bytes originally contained in the file. This gives you an idea of the extent of data loss.

The recovered file is stored under its original file name in the directory in which it originated. Bad sectors are marked so as not to be accessed again by the system. When the recovery operation is complete, you may access the file by using normal procedures, reedit and correct it as required, and use it in normal system and program functions.

Remember that you should not use wildcard characters in the file specification for the file to be recovered. When the RECOVER command is entered with the *filespec* parameter, only one file will be recovered. If you enter a file specification that includes wildcard characters, only the first file found that matches the specification will be recovered.

Recovering an Entire Disk

If a disk contains many bad sectors or if a directory has been damaged, you can recover all files on the disk by entering RECOVER followed by the drive name of the drive in which the disk is located. For example, suppose you have a damaged data disk in drive C and wish to recover as many of the files from the disk as possible. You could do this by entering

```
RECOVER C:
```

and pressing **RETURN**. The screen would display

```
Press any key to begin recovery of the
file(s) on drive C
```

If you want to abort the RECOVER operation at this point, you may do so by pressing **CTRL-BREAK**. The system prompt will be displayed.

If you want to continue with the RECOVER operation, press any alphanumeric key (that is, any nonfunction or noncontrol key) and the recovery operation will begin. RECOVER will read the disk sector by sector, marking bad sectors and writing the recovered files to the root directory. (RECOVER does not recover the hierarchical directory structure that may exist on MS-DOS version 2 disks.) When the recovery operation is complete, the screen displays

```
nn file(s) recovered
```

where *nn* is the number of files recovered, followed by the system prompt.

If you know how many files were originally on the disk, this message will give you an idea about the extent of data loss.

Recovered files are sequentially assigned file names in the form FILE*nnnn*.REC, where *nnnn* is a four-digit number; the series of numbers used in the file names assigned by RECOVER begins with 0001. Thus, if you were to display a directory of the recovered disk from the above example, the screen might display the following directory:

```
 Volume in drive C has no label
 Directory of C:\

FILE0001 REC    13312   9-13-83  11:01a
FILE0002 REC    17408   9-13-83  11:01a
FILE0003 REC     1024   9-13-83  11:01a
.
.
.
FILE00nn REC    16384   9-13-83  11:01a

        nn File(s)  249856 bytes free
```

Notice that the date and time of file creation is the date and time that the *recovered* file was created. In addition, the size displayed in the directory for a given recovered file may not coincide with the size of the file before recovery; the recovered file may include more or fewer bytes and will probably include added information or unused space at the end of the file as a result of recovery. This is because the size of a recovered file is a multiple of the MS-DOS allocation unit size.

You can access the recovered files by using normal procedures to reedit and correct them as required, rename them (having identified them by their contents during edit), and use them in normal system and program operations.

Unless an entire disk is badly damaged, it is generally better to use RECOVER to regain individual files than to use RECOVER to regain all files on the disk. Because RECOVER has no way to check whether the data in the disk's directory is valid or invalid, it recovers all files as described above, including files for which there may still have been valid directory entries.

Error Message

```
Invalid number of parameters
```

EXPLANATION: This message is displayed if you entered the RECOVER command without specifying either a file specification or a drive name for recovery. Reenter the RECOVER command line, specifying the file or drive name that you want to recover.

REM (Resident, Batch-Processing)

Purpose

Used to display user-created comment or remark. Often used in batch files.

Entry Form

```
REM [remark]
. [remark]
```

where ***remark*** is an optional user-defined remark or comment.

Preliminary Concepts

The resident, batch-processing commands are most often executed from within a batch file, although they may be used directly from a command line at the system prompt in some instances. Refer to the section Automatic Command Entry in Chapter 5, "Command Features," for a complete explanation of batch file creation and execution.

NOTE: You must always end each line within a batch file by pressing the RETURN key.

Command Line Entry

This command is invoked simply by entering **REM** or **.** at the system prompt and pressing **RETURN**. The REM command may be used within batch files without an optional *remark* to provide spacing for greater readability. Whenever this command is used with the optional *remark*, the *remark* must be on the same line as the command. The *remark* will be displayed when the command is executed. The only valid separators that may be used within the *remark* are a space, tab, or comma.

If you wish to display a multiline remark, you must use the REM command at the beginning of each line.

Using the REM Command

When the system encounters the REM command, any remark or comment on the same line as the command is displayed on the screen. This enables you to put informational or instructional messages in a batch file for the user's convenience. If the command is entered in a batch file without the optional remark, you have in effect created a blank line within the batch file. This may be used to enhance batch file readability. (For information on creating and using batch files, refer to Chapter 5, "Command Features.")

REM

Other than creating a blank line or displaying a message on the screen, the REM command has no effect on processing.

By using the REM command, you can create a batch file and easily remind yourself at a later time what the file does. For example, assume that you create a batch file to prepare new disks for use in your system. The batch file you create might look like the following example if you use the REM command to annotate the file.

```
REM  This file checks new disks
REM  It is named NEWDISK.BAT
PAUSE Insert new disk in drive B:
Format B:/S
DIR B:
CHKDSK B:
```

With this batch file, the first two lines are displayed on the screen as soon as the batch file name (NEWDISK) is entered and RETURN is pressed. The next line causes a display instructing the user to insert a new disk in the appropriate drive. Commands in the remainder of the batch file format the disk, transfer system files to the newly formatted disk, display the directory (enabling the user to verify that the previous operation was completed), and execute the CHKDSK command to verify disk validity.

You may also use REM to create an audit trail for others who may use a given batch file. Another user could display the file with a text editor, EDLIN, or the TYPE command, read the command entries in the file along with the remarks, and understand the batch file's use and structure.

In general, remarks are helpful when inserted in a batch file to display instructional or informational messages about ongoing batch operations. The REM command enables you to create your batch files to guide or prompt the user in this way.

REN or RENAME (Resident)

Purpose

Use to change the name of an existing file.

Entry Forms

```
REN filespec filename
RENAME filespec filename
```

where ***filespec*** is the file specification identifying the file you wish to rename, and
filename is the new file name you wish to give the specified file.

Command Line Entry

This command can be invoked by entering either REN or RENAME at the system prompt. The required parameters of the RENAME command line are described below. Both parameters must be entered and are user-defined.

Existing File Specification

The first parameter that must be entered following the RENAME command is the *filespec* of the existing file. This *filespec* identifies to the system the file that is to be renamed. The *filespec* may be a full file specification (drive designation, file name, and extension) or only a file name (and extension, if any) if the file to be renamed is in the current directory of the default disk.

A drive name must be entered as part of the file specification if the file is not on the disk in the default drive. Additionally, You can also use wildcards to change only certain letters in a single file name. For example, you could use the following command to rename the file ABODE on drive B: to ADOBE:

```
REN B:ABODE ?D?B?
```

Refer to Chapter 1 for more information on wildcard characters.

Error Message

```
Duplicate file name or File not found
```

EXPLANATION: This error message may be displayed if:

- you attempted to rename a file to a name that already exists in the current directory,
- the file you specified to be renamed (in other words, the first parameter entered following the REN command) does not exist in the current or specified directory, or
- you attempted to rename a directory.

Whenever this message is displayed, the renaming operation is aborted and the system prompt is displayed. You must reenter a valid REN command at the system prompt.

RESTORE (Transient)

Purpose

Use this command to restore or recreate the individual files contained in a backup file created with BACKUP.

Entry Forms

```
RESTORE
RESTORE ?
RESTORE [[d:]filename [filespec[+filespec...]]][/x...]
```

where ***d*** is the drive name of the source drive in which the backup file that contains the file(s) to be restored is located;
filename is the primary file name of the backup file;
filespec is the original file specification for a file or group of files to be restored from the backup file; and
/***x*** is one or more of the following switches:

/A[:*date*]	After date—Restore files stamped after the specified date.
/B[:*date*]	Before date—Restore files stamped before the specified date.
/D	Directory master—Locate all master backup files and give directory.
/E	Exception files—Exclude listed files from the restoration operation.
/F	Flat restoration—Restore all files to the current directory, regardless of the directory level in which they originated.
/L	List directory—List internal directory of the backup file.
/M:*dd*	Map output drive—Restore files to the specified drive instead of the original drive.
/O	Overwrite files—Overwrite existing files on the destination disk.
/Q	Query each—Query "yes" or "no" on each file before restoration.
/R	Review selected files—Show list of files that will be restored before beginning operation.
/T	Today's date—Restore files with today's date.
/V	Verify files—Verify recreated files after restoration.

Preliminary Concepts

As BACKUP provides a method for you to automate backing up routine work, RESTORE automates the retrieval of data stored in backup files. RESTORE unlinks the sequential file structure of a backup file (created during BACKUP) and restores the individual files back to their original devices. Like BACKUP, RESTORE supports a number of optional switches that provide you a large degree of control over the restoration of files. (For information on the BACKUP command, refer to the BACKUP section of this chapter.)

The RESTORE command may be invoked in two ways: *interactive entry*, wherein RESTORE prompts you for the required input; or *command line entry*, wherein you input the required parameters when you invoke the command. In addition, RESTORE provides a help screen display that briefly describes the RESTORE command and lists its syntax requirements and supported switches.

Normal events that occur during execution of the RESTORE command are prompts regarding the exchange of backup file disks (where the backup file is stored on more than one disk volume), a screen display of the file specification for each file that is restored, and a prompt regarding the existence of duplicate file names on the source and destination disks.

If the backup file you are restoring is on more than one disk volume and RESTORE needs a volume other than the one currently in the source drive, execution stops temporarily and the screen displays

```
Insert volume nnn, filename.nnn-1, in drive d and press
RETURN when ready.
```

In this prompt, *nnn* is the volume number that BACKUP assigned to the disk containing backup file *filename.nnn-1*. (Notice that the volume number is always one more than the numeric extension assigned to the backup file's name.) Insert the disk volume requested in the drive (*d*) indicated, and press **RETURN**. The restoration operation will resume.

During execution of RESTORE, the screen displays the file specification of each file as it is restored. This listing takes the following form:

```
A:FILENAM1.EXT
B:FILENAM2.EXT
B:FILENAM3.EXT
.
.
.
B:FILENAMn.EXT
```

Notice that in the listing of files restored, the drive name provided for each file is the drive to which the restored file is being written. This may not be the same as the original source drive name (the drive from which the file was backed up) if the /M (Map output drive) switch was used when RESTORE was invoked.

Occasionally you may encounter the following message when executing RESTORE:

```
File filespec already exists, do you wish to delete it (Y/N) ?
```

When this message is displayed, RESTORE has encountered a file on the destination disk that has the same primary file name and extension as a file that is to be restored. If you want RESTORE to overwrite the existing file with the restored file, press **Y**. If you do not want RESTORE to overwrite the existing file, press **N**. If you want RESTORE to overwrite any existing files automatically without displaying the above prompt, use the /O (Overwrite files) switch when you enter a RESTORE command line.

RESTORE Help Screen

To obtain a screen display that summarizes the use, syntax requirements, and switches of the RESTORE command, enter **RESTORE ?** at the system prompt and press **RETURN**. The help screen is displayed as shown in Figure 11.17. The display is followed immediately by the system prompt, enabling you to refer to the information on the screen as you enter a RESTORE command line. More information on the various switches shown in Figure 11.17 is provided under Command Line Entry in this section.

```
              RESTORE version 2.0



The  RESTORE  utility is designed to complement the BACKUP
utility.  RESTORE extracts specified files from the single
long file that BACKUP creates; then it returns those files
to their original drives.

Syntax: A:RESTORE [[<d:>]<filename> [<filespec>[+<filespec>...]]][</x>...]

     Switches: Default state: /O off, /Q off and /V off.

/A AFTER date.  /A:<mm-dd-yy>          /M MAP output drive. /M:lp
/B BEFORE date. /B:<mm-dd-yy>          /O OVERWRITE files.
/D DIRECTORY master.                   /Q QUERY each.
/E EXCEPTION files. /E:<filespec>      /R REVIEW selectedfiles.
/F FLAT restoration.                   /T TODAY's date.
/L LIST directory. <filespec>/L        /V VERIFY files.


A>
```

Figure 11.17. RESTORE Help Screen Display

Interactive Entry Prompt and Response

If you enter **RESTORE** and press **RETURN** without entering any other information, the system will display the RESTORE banner followed by the RESTORE command prompt > as shown in Figure 11.18. When the command prompt is displayed, you may enter commands to RESTORE. When one command line has been executed, the command prompt will be displayed again and you may enter another command line. RESTORE terminates *only* when you press RETURN or CTRL-BREAK at the command prompt without having entered a command line.

The command line entries that may be made at the RESTORE command prompt are the same as those described under Com-

```
          RESTORE version 2.0

>
```

Figure 11.18. RESTORE Banner and Command Prompt

mand Line Entry in this section, with the exception that you do not need to reinvoke RESTORE by beginning each line with the command name.

Command Line Entry

If you enter a complete, valid RESTORE command line at the system prompt and press RETURN, the single command you entered will be executed and then the system prompt will be displayed again. If you wish to complete another restoration operation, you must reinvoke the RESTORE command, unlike when using the interactive method described above.

You must always begin the command line with RESTORE and follow it with the required parameters and desired optional switches. The parameters of the RESTORE command line are described below. Entry requirements are noted, where appropriate, in the parameter descriptions.

Source File Specification

You must always enter one source *filename* following **RESTORE** (or following the command prompt if you use the interactive entry method). The source *filename* must be the name of a backup file that was created with BACKUP and must not include any wildcard characters. Do not enter an extension for the source file name; RESTORE assumes the three-digit extensions automatically assigned by BACKUP, wherein .000 is the master backup file, .001 is the second volume, and .002 is the third, and so on.

If the backup file that you wish to restore is not on the disk in the default drive, you must precede the source *filename* with the appropriate drive name (*d*).

Destination Files

Following the source file specification, you must enter at least one destination file specification or *filespec* to describe which files in the backup file (*filename*, as described under Source File Specification) are to be restored. Wildcard characters (* and/or ?) may be used in the destination *filespec* to designate more than one file.

You may also enter a series of file specifications, as in *filespec*[+*filespec*...]. When a series of destination file specifications is entered, each *filespec* in the series must be separated from the adjacent file specification(s) by a plus sign (+). There should be no spaces or delimiters other than the plus sign between the *filespecs*. Some or all of the file specifications in the series may include wildcard characters. Destination files may be on the same or different disks, or in the same or different directories.

The destination specifications that you enter for this command are unique in MS-DOS in that they describe (or further define) the source as well as specify the destinations for command output.

Normally, RESTORE automatically directs the files from the backup file to the drive on which they originated (that is, the drive in which they were located when you ran BACKUP). For this reason, you should always include the drive name in the destination file specification unless the files were backed up from the default drive. If you do not remember the drives from which the files were backed up, you can obtain an internal directory of the backup file by using the /L switch with either the RESTORE or the BACKUP command. (Refer to the information provided under Switches below, or to the BACKUP section of this chapter.)

Switches

The use of any of the switches (/*x*) supported by the RESTORE command is optional. They are provided to allow you to tailor execution of the command to meet your needs. The switches are entered at the end of the command line and do *not* need to be separated (either from the preceding input or from each other) by a space. The switch character (/) is the only delimiter required.

If no switches are entered, RESTORE assumes the following operational defaults:

- query for overwrite of existing files (may be changed with /O switch),
- do not query for each file name before restoration (may be changed with /Q switch),
- do not verify files after restoration (may be changed with /V switch), and
- do a full restoration, including restoration of original directory structure (may be changed by using the /F switch).

Supported switches are described individually below.

/A—After Date

Use the /A switch to restore all files that match the destination file specification(s) that are dated (in the original directory) *after* today's or the specified date. (The date recorded in the directory for any given file is called that file's *date stamp.*) If you wish to restore files with a date stamp later than the current (today's) date displayed by the DATE command, enter the switch as

```
/A
```

at the end of the command line. All files dated after the date given to MS-DOS either when it was first booted up or when the DATE command was last used will be restored from the backup file. (RESTORE compares the original directory date stamp for the specified file(s) with the date established by the DATE command.)

If you wish to specify a date other than the current date, enter the switch by using the format

```
/A:mm-dd-yy
```

where *mm* = a number from 1 to 12, signifying the month;
dd = a number from 1 to 31, signifying the day of the month; and
yy = a number from 00 to 99, signifying the year.

Note that the switch and the specified date must be separated by a colon (:), and that the only valid separator for the numbers in the specified date is a hyphen (-).

When the date is specified with the /A switch, RESTORE compares the directory date stamp for the specified destination file(s) to the date you entered and restores all files with a date later than the specified date.

/B—Before Date

Use the /B switch to restore all files that match the source file specification(s) that are dated (in the original directory) *before* today's or the specified date. If you wish to restore files with a date stamp earlier than the current (today's) date displayed by the DATE command, enter the switch as

```
/B
```

at the end of the command line. All files dated before the date that was given to MS-DOS either when it was first booted up or when the DATE command was last used will be restored. (RESTORE compares the directory date stamp for the specified file(s) to the date established by the DATE command.)

If you wish to specify a date other than the current date, enter the switch using the format

```
/B:mm-dd-yy
```

where ***mm*** = a number from 1 to 12, signifying the month;
dd = a number from 1 to 31, signifying the day of the month; and
yy = a number from 00 to 99, signifying the year.

Note that the switch and the specified date must be separated by a colon (:) and that the only valid separator for the numbers in the specified date is a hyphen (-).

When the date is specified, RESTORE compares the directory date stamp for the specified source file(s) to the date you entered and restores all files with a date earlier than the specified date.

/D—Directory Master

Use the /D switch alone to check the default or a specified disk for *master backup files* and to display a master backup file directory. A master backup file is one with the extension ".000." In addition to file copies, the master backup file includes the number of volumes that the backup file resides on, the number of files (sources) the backup file contains, and the date it was made.

This switch must be entered using the command line entry form

```
RESTORE [d:]/D
```

if the command is invoked at the system prompt; or the form

```
[d:]/D
```

if the command is invoked at the command prompt >. In either case, *d* is the drive name of the desired drive (if you want a master backup file directory for a disk other than that in the default drive). When you press **RETURN**, the screen displays a master file directory similar to that shown in Figure 11.19. The date in the upper right corner of the display is the current date; all other information in the display pertains to the master backup files that the system locates.

```
Name       Volumes   Files     Date      4-15-84

BACKFILE      3        10     12-19-83
BACK2         5       127      9-01-83
BACK3         1         1      8-22-83
```

Figure 11.19. Typical Master Backup File Directory

/E—Exception Files

Use this switch to specify a file or files to be *excluded* from the restoration operation. Exception files are listed immediately following the switch, in the form

/E:*filespec*[+*filespec*...]

Note that a colon (:) *must* be used between the switch (/E) and the exception file(s) listed. Wildcard characters may be used in some or all of the exception file specifications. When more than one exception file is listed, the file specifications must be separated by the plus sign (+). No spaces or delimiters other than the plus sign should be used between the *filespecs*.

When the /E switch is used, all files not listed after the switch that *do* match the destination file specification(s) will be restored. This switch is particularly useful when you want to restore many files, have used wildcard characters in the destination file specification(s), and wish to exclude some files from the restoration operation. An example is provided under Using the Exception Files and Query Each Switches in this section.

/F—Flat Restoration

Normally, RESTORE reads directory information stored in the backup file, recreates original directories and subdirectories, and replaces files in their original directory levels. Use the /F switch to restore files from multiple directory levels (when BACKUP was executed using the /G switch) to a single directory level. The /F switch (signifying "flat") eliminates the original directory tree structure and places all restored files in the current directory of the destination disk, regardless of the directory in which they were originally located.

/L—List Directory

This switch enables you to display a directory of the files that are contained within a specified backup file. The switch must be entered as part of a command line having the form

```
RESTORE [d:]filename/L
```

if entered at the system prompt, or

```
[d:]filename/L
```

if entered at the RESTORE command prompt >. In either case, *filename* is the primary file name of the backup file for which you wish to obtain an internal directory listing. The drive name (*d*) must be entered if the backup file is on a disk that is in a drive other than the default. Do not enter a file name extension for the backup file; RESTORE assumes the extension ".000."

When you enter a command line that includes the /L switch and then press RETURN, the screen displays a listing of the individual files contained in the backup file, the drive from which each file was backed up, the size of each file in bytes, the disk volume numbers on which the files are stored (one file may be stored across more than one disk volume), and the original directory date stamp for each file.

The display also includes entries for subdirectories (if any) and their parent directories. These are identified by <DIR> in the Start Volume column of the display. (Subdirectories will be included in a backup file *only* if the /G switch was used when the backup file was created. Refer to the BACKUP section of this chapter for more information.)

A typical internal directory listing is shown in Figure 11.20. Notice that a summary line at the bottom of the display gives the total number of files contained in the backup file, and the total number of volumes on which the backup file resides. Also, notice that the subdirectory (DATA in Figure 11.20) and the entry for its parent directory (..) are *not* counted as files in the backup file internal directory. The files that originated in the subdirectory are listed between the subdirectory name and the entry for the subdirectory's parent.

/M—Map Output Drive

Normally, RESTORE places files on the disk in the drive from which they were originally backed up, as identified by the drive name listed in the backup file's internal directory. (See the Drive field in Figure 11.20.) Use the /M switch to direct the restored files to a disk other than the original disk.

When you use the /M switch, you must specify both the original (source) drive name and the new destination drive name in order for RESTORE to redirect its output. The switch must be entered at the end of the command line in the form

`/M:`***dd***

```
Drive     File Name      Date    Start    End    Size in
                                 Volume  Volume   bytes

E:      TESTFIL1.DAT  10-10-82    1        1       3264
E:      TESTFIL2.DAT  10-10-82    1        2      19582
F:      TESTFIL1.DOC   8-01-82    2        3     887236
F:      TESTFIL2.DOC   9-17-82    3        3      22230
F:      DATA          10-10-82  <DIR>
F:      EXP1.DAT      10-10-82    3        3       1548
F:      EXP2.DAT      10-15-82    3        3       1675
F:      ..            10-10-82  <DIR>

6 file(s) on 3 volume(s)
```

Figure 11.20. Typical Backup File Internal Directory Listing

where the first *d* is the original source drive and the second *d* is the desired destination drive. Notice that the switch and drive names must be separated by a colon (:), but that the drive names themselves are not separated, even by a space. For example, if you wish to restore files backed up from drive E to a RESTORE destination disk in drive A, you would enter the switch as

```
/M:EA
```

You may use more than one /M switch in a single command line to effect more than one output mapping. You may wish to do this when you are restoring files originally copied from several source drives and you wish to write all the files to one disk.

/O—Overwrite Files

Normally, if RESTORE encounters a file on the destination disk that has the same file name as a file that is being restored, execution is suspended as the system prompts

```
File <filespec> already exists, do you wish to delete it (Y/N) ?
```

If you wish to save the existing file, you may press **N** and RESTORE will not overwrite it. (Neither will the file from the backup disk be restored; it will be skipped and RESTORE will go on to the next file in the backup file.) If you want RESTORE to overwrite the existing file with the file from the backup disk, press **Y**.

Use the /O switch to cause RESTORE to overwrite existing files automatically, without prompting you when a match in file names is encountered. You may wish to use this switch when you are restoring a disk or selected group of files to an old disk that you have used before, but that is not as current as your backup file.

/Q—Query Each

When you use this switch, RESTORE will locate all files that match the destination file specification(s) and prompt

```
Restore filespec (Y/N) ?
```

where *filespec* is a unique file specification for each file before beginning the restoration operation.

The /Q switch is especially useful when you use wildcard characters in the destination file specification and/or when you list a series of destination files, because you are given the opportunity to review the list of files that will be restored and delete the files (if any) that you do not want to restore. You cannot, however, add any files to the destination files specified. If you have inadvertently omitted the file specification for a file that you wish to restore, you will have to invoke a second RESTORE operation when the first is completed.

If you *do* wish to restore a given file, press **Y** at the `Restore` *`filespec`* `(Y/N) ?` prompt for that file.

If you *do not* wish to restore a given file, press **N** at the prompt for that file; the file will be deleted from RESTORE's internal list of files to restore before the restoration operation is begun.

/R—Review Selected Files

This switch is similar to the /Q switch, in that it enables you to review the files selected for restoration before the restoration operation is begun. However, the /R switch lists the entire group of files matching the destination file specification(s) at one time, and you must accept or reject the entire group rather than individual files.

When you use the /R switch, RESTORE locates all files that match the destination file specification(s), and then displays

```
                    RESTORE version 2.0


Files to be restored are:
```

followed by a list of the file specifications for files that will be restored. Just below the list, the system prompts

```
Is this correct (Y/N) ?
```

If the list is correct (that is, if it correctly shows all files that you wish to restore and does not include any files that you do not wish to restore), press **Y**. The restoration operation will begin.

If the list is not correct, press **N**. The restoration operation will be aborted and the system or command prompt will be displayed.

/T—Today's Date

Use this switch to restore all files that match the destination file specification(s) and are dated with the current date (as displayed by the DATE command). When you enter /T as part of the command line, RESTORE compares the original directory date stamp for all specified file(s) to the date established by the DATE command. Then, only those files that have a matching date are restored.

/V—Verify Files

Use this switch to cause RESTORE to verify the accuracy of each destination file immediately after it is copied from the backup file. When you enter /V as part of the command line, RESTORE compares each destination file with the source (backup) file copy to make sure that there are no discrepancies. The file specification of each destination file is displayed during its verification in the form

```
filespec
Verifying filespec
```

If (and only if) an error is found, RESTORE prompts

```
Verify error, try RESTORE again (Y/N) ?
```

If you wish to reattempt restoration and verification for the file, press **Y**. The restoration and verification operations will be repeated. If you do not wish to reattempt the restoration operation, press **N**. The restoration operation for that file will be aborted, and RESTORE will go on to the next specified file.

If no verification errors are found, the screen simply displays the names of the files being restored and verified.

Specifying a Series of Destination Files

Just as you can tailor BACKUP to back up only a specified selection of files, you can execute RESTORE for selected files within a backup file and not have to restore every original file. In the BACKUP section of this chapter, an example was provided for backing up a series of files from different disks to the backup file BIGFILE on the default disk. The example command line was

```
BACKUP A:*.BAS+B:*.DOC+B:???X.COM+E:*.*BIGFILE
```

Now, suppose you wish to restore only the .BAS files originally from the disk in drive A and the .COM files originally from the disk in drive B. With disks available in drives A and B and RESTORE and BIGFILE available on the disk in the default drive, you could restore the selected files by entering

```
RESTORE BIGFILE A:*.BAS+B:*.COM
```

and pressing **RETURN**. As an alternative, you could enter the command line **RESTORE BIGFILE A:*.*+B:*.COM** since the only files copied from the disk in drive A were .BAS files.

If you wished to restore all files from the backup file to disks in the original source drives, you could enter the series of destination file specifications in the command line as follows:

```
RESTORE BIGFILE A:*.*+B:*.*+E:*.*
```

When you press **RETURN**, RESTORE will locate all files within the backup file that were from drive A and write them to drive A, all files from drive B and write them to drive B, and all files from drive E and write them to drive E. Note that it is *not* necessary to specify the destination files exactly as they were specified as source files for BACKUP; it is only necessary to enter the file specifications in a form that enables RESTORE to locate the correct file(s) within the backup file. You must always enter the correct drive name as it appears in the internal directory listing for the backup file. Then, if you do not want a file or group of files restored to a disk in the original source drive, use the /M (Map output drive) switch.

Using the Exception Files and Query Each Switches

Suppose you have a backup file named MISCBAK that contains backup copies of correspondence files originally from a disk in drive D—and that the files were named with standardized extensions enabling you to identify to whom the correspondence was sent. Further, suppose that the file name extensions used were .CHI for letters sent to a Chicago office, .STJ for letters sent to a St. Joseph office, .VEN for letters sent to vendors, and .PER for personal letters.

If you wished to restore the files for all business correspondence but did not wish to restore the files for personal correspondence, you could enter

```
RESTORE B:MISCBAK D:*.*/E:D:*.PER
```

and press **RETURN**. All files with the extensions .CHI, .STJ, and .VEN that were backed up from drive D would be restored.

You could also use the /Q (Query each) switch to selectively restore some correspondence files. Assuming the same circumstances as above, you could enter

```
RESTORE B:MISCBAK D:*.*/Q
```

and press **RETURN**. In this case, RESTORE would prompt you

```
RESTORE D:filename.ext (Y/N) ?
```

for each file in the backup file (MISCBAK) before the restoration operation was begun.

If you wished to restore the file named in the prompt, you would press **Y**.

If you did not wish to restore the file named, you would press **N**.

When you had responded to the prompt for all files in the backup file, restoration of the files for which you pressed **Y** would begin.

Using the Interactive RESTORE Method

If you have a backup file that contains a variety of files from a number of source disks that you wish to restore selectively, you may find it most convenient to use RESTORE by the interactive method. For example, suppose the backup file STUFF contains letter, document, and data files with the extensions .LTR, .DOC, and .DAT, respectively. Also assume that the files were originally from disks in drives D and E. If you wish to restore all letter files to one disk, all document files to another, and all data files to a third disk, proceed as follows:

NOTE: This example assumes that RESTORE and the backup file are both available on the default drive.

- At the system prompt, enter **RESTORE** and press **RETURN**. The RESTORE command prompt > will be displayed.
- To restore all letter files from the backup file to the disk in drive D, enter

```
STUFF D:*.LTR+E:*.LTR/M:ED/V
```

and press **RETURN**. RESTORE finds all .LTR files originally from drive D within the backup file and writes them to the disk in drive D. RESTORE then finds all .LTR files originally from drive E and writes them to the disk in drive D; the /M switch redirects the files originally from a disk in drive E to drive D. The /V switch causes RESTORE to verify that all files were copied accurately to the destination disk. When all .LTR files have been restored, the command prompt > is displayed again.

- To restore all document files from the backup file to the disk in drive E, enter

```
STUFF D:*.DOC+E:*.DOC/M:DE/V
```

and press **RETURN**. RESTORE finds all .DOC files originally from drive D within the backup file and writes them to the disk in drive E; the /M switch redirects the files originally from a disk in drive D to drive E. RESTORE then finds all .DOC files originally from drive E and writes them to the disk in drive E. The /V switch causes RESTORE to verify that all files were copied accurately to the destination disk. When all .DOC files have been restored, the command prompt is displayed again.

- To restore all data files from the backup file to a disk in drive D, insert a new blank disk in drive D and then enter

```
STUFF D:*.DAT+E:*.DAT/M:ED/V
```

and press **RETURN**. RESTORE finds all .DAT files originally from drive D within the backup file and writes them to the disk in drive D. RESTORE then finds all .DAT files originally from drive E and writes them to the disk in drive D; the /M switch redirects the files originally from a disk in drive E to the disk in drive D. The /V switch causes RESTORE to verify that all files were copied accurately to the destination disk. When all .DAT files have been restored, the command prompt is displayed again.

- To exit RESTORE and return to the operating system, simply press **RETURN** when the command prompt is displayed.

Error Messages

```
Backup file name cannot be ambiguous.
```

EXPLANATION: This message occurs if the wildcard characters * and/or ? occur in the backup file's name (that is, the source file specification).

```
Cannot find master backup file filename.000.
```

EXPLANATION: This message occurs when you have requested an internal directory listing (by using the /L switch in a RESTORE command line) for a given backup file (*filename*.000) that does not exist on the default or specified disk. Make certain that the correct disk is available and reenter the command line with the appropriate drive name.

```
Cannot open backup file d:filename.nnn,
insert another disk and press RETURN,
or press any other key to abort.
```

EXPLANATION: This message occurs when you are asked to insert volume nnn+1 (which would contain backup file *filename.nnn*) and the wrong disk is inserted. Insert the correct disk and press **RETURN** to continue.

```
Cannot open master backup file d:filename.000,
insert another disk and press RETURN,
or press any other key to abort.
```

EXPLANATION: This message occurs if the disk that has been inserted is not volume 1 of backup file *filename*. Insert the correct disk and press **RETURN** to continue.

```
Extension on backup file specified will be ignored.
```

EXPLANATION: This message occurs whenever you try to specify a backup file extension. When the RESTORE operation begins, it always requires the backup master file with the extension .000. If you enter a backup file name extension, RESTORE ignores it and uses the standard, sequentially numbered extensions.

```
File filespec already exists, do you wish to delete it (Y/N) ?
```

EXPLANATION: This message occurs if the /O (Overwrite files) switch has not been used during a RESTORE operation. It means that RESTORE has encountered a file on the destination disk that has the same name as the one that is about to be restored.

If you want the existing file overwritten by the restored file, press **Y**.

If you do not want the existing file to be overwritten, press **N**. The existing file on the destination disk will be retained; the file in the backup file that has the same name will *not* be restored.

```
File filespec not found.
```

EXPLANATION: This message occurs whenever a file is specified for restoration and that file is not on the backup disk. Check whether you made an error in entering the file specification (by checking the internal directory of the backup file if necessary), and reenter a RESTORE command line for the file if an error was made.

```
Invalid backup file.
```

EXPLANATION: This message occurs if the backup file specified in a RESTORE command does not contain valid information. This may occur if the file specified was not a backup file but had a .000 extension, or if the data in a backup file has degraded (possibly due to a bad sector, or inadvertent exposure to an electromagnetic field).

`Invalid date in switch.`

EXPLANATION: This message occurs if the date given with the /A or /B switch was not in the correct format. Reenter the command line including a valid date in the proper syntax.

`Invalid drive designation on RESTORE file.`

EXPLANATION: This message occurs when a drive name is used that is not in the range of supported names (A through H) or that does not exist in your system.

`Invalid exception file specifications.`

EXPLANATION: This message will occur if you failed to specify at least one exception file after the /E switch or if exception files were specified with a syntax error. Reenter the RESTORE command line with the required /E switch parameter(s) in the correct syntax.

`Invalid file name.`

EXPLANATION: This message appears when a file name is specified that does not conform to the MS-DOS file naming conventions. Reenter the command line using a valid file name.

`Invalid selection file specifications.`

EXPLANATION: This message is generally caused by a typographical error in the command line. The message results when parameters in the command line appear garbled or are incorrectly punctuated.

Invalid switch /*x* specified.

EXPLANATION: This message occurs if RESTORE is unable to recognize the switch that was specified in the command line.

Invalid version of RESTORE for file *filename*.000.

EXPLANATION: This message occurs only if you try to invoke a RESTORE function using a backup file created with an incompatible version of the BACKUP program.

```
No files selected.
```

EXPLANATION: This error message occurs if the destination file specifications were not files that were contained in the backup file, or if the destination file specifications were overruled by file specifications in command line switches.

```
Not enough parameters specified.
```

EXPLANATION: This message results when the command to RESTORE is not complete enough for RESTORE to carry out the intended operations. Refer to Command Line Entry in this section, and reenter a valid RESTORE command line.

```
Out of disk on restoration of filename, insert another disk
and press return, or press any other key to abort.
```

EXPLANATION: This message occurs during a restore operation if the destination disk becomes full. To continue restoration of the backup file, insert a blank, formatted disk in the destination drive and press **RETURN**.

```
Too many parameters specified.
```

EXPLANATION: This message will appear if you have entered more parameters than RESTORE can handle properly. Refer to Command Line Entry in this section and reenter a valid RESTORE command line.

```
Verify error, try restore again (Y/N) ?
```

EXPLANATION: This message occurs if the files did not verify correctly after a restore operation. You can choose to repeat the restoration of the file that could not be verified, or to advance to the restoration and verification of the next file within the backup file.

If you wish to retry the restoration and verification operation for the file, press **Y**.

If you wish to skip the file in which the verification error occurred, press **N**. The restoration operation will be begun for the next specified file.

RMDIR or RD (Resident)

Purpose

The RMDIR (remove directory) command is used to remove a directory from the directory structure.

Entry Forms

```
RMDIR [d:]pathname
RD [d:]pathname
```

where ***d*** is the letter of the designated disk drive; and
pathname is a sequence of characters of the form:

```
[\] [directory] [\directory...]
```

Preliminary Concepts

This command removes a directory that is empty except for the '•' and '••' MS-DOS shorthand symbols.

You cannot delete • because that is the name of the current working directory (that is, the one you are currently in). You cannot delete •• because that is the *parent* of the current working directory. This is because the current working directory exists *in* the •• directory.

NOTE: The two shorthand symbols do not exist in the root.

Any directory you wish to remove with the RMDIR command *must* be empty except for the • and •• symbols. This is to prevent you from accidentally deleting directories and files. You can remove any directory, except the root, by specifying its path name to the RMDIR command.

To delete all the files in a directory (so that you can remove the directory), simply type DEL followed by the path name of the directory, for example

```
DEL \USER
```

and press **RETURN**, and all the files in subdirectory USER will be deleted. (Refer to the section on the DELETE command for more information.)

NOTE: You cannot delete the • and •• entries or directories with the DELETE command.

Command Line Entry

If you want to remove a directory, you would first issue a DIR command for that directory to make sure that the directory does not contain any important files that you do *not* want deleted.

Having made sure that the directory is empty, enter

```
RMDIR \BIN\USER\JOE
```

and press **RETURN**. The directory will then be deleted from the hierarchical directory structure.

The previous example was executed from the root directory. You could also delete the directory \BIN\USER\JOE, from the parent directory \BIN\USER by entering

```
RD JOE
```

and pressing **RETURN**.

Error Messages

```
Invalid number of parameters
```

EXPLANATION: This error message will be displayed if you do not enter the proper number of command parameters after RMDIR. For example, if you enter

```
RMDIR
```

and press **RETURN**, MS-DOS will display this error message. To correct the situation, reenter the command line with the proper number of parameters.

```
Invalid path, not directory,
or directory not empty
```

EXPLANATION: This error message will be displayed under three different circumstances.

- If you try to remove a directory while in that directory's parent and you specify a \ before the directory's name, you will get this error. That is because you specified an invalid path.
- If you try to remove a file instead of a directory, with this command, you will get this error.
- If you attempt to remove a directory that contains files, you will also get this error.

SEARCH (Transient)

Purpose

The SEARCH command is used to locate files within the hierarchical directory structure.

Entry Form

`SEARCH[ afn] [/x]`

where ***afn*** is an ambiguous file name that may include the wildcard characters, * or ?. The *afn* variable may also be a *filespec*, including a drive name, or a *pathname*; and

/x can be any of the following switches:

/C is a switch that causes SEARCH *not* to search subdirectories;

/D is a switch that causes SEARCH to list the names of directories while it is searching; and

/T is a switch that causes SEARCH to display a graphic representation of the directory structure.

Preliminary Concepts

SEARCH locates files matching the specified *afn* and outputs their full path names to STDOUT (standard output, normally your screen). This can be very helpful when used in conjunction with programs such as APPLY. SEARCH will search all subdirectories of the current directory (except when the /C switch is specified). The names of the directories are not listed (unless the /D switch is specified).

Command Line Entry

SEARCH is entered by typing

`SEARCH[ afn]`

and pressing **RETURN**. In addition, any of the following switches may be entered:

Using the /C switch causes SEARCH *not* to search subdirectories, but only the current directory, for matching files.

SEARCH

Using the /D switch causes SEARCH to list the names of the directories as it searches. This can be useful for keeping track of how the search is progressing when the /T switch is not used.

Using the /T switch causes SEARCH to alter its output to a graphic representation of the directory structure. In this mode, directories are listed as they are encountered, with each level of the directory structure indicated by an indentation of four spaces. (That is, the /T switch displays the tree-like structure of your files.)

The SEARCH command also gives you the option of tracing a number of path names by specifying the asterisk (*) wildcard character.

If you wish to find the path names for a number of files with the same file extension (.ext), you would enter

```
SEARCH *.ext
```

and press **RETURN**.

where ***ext*** is the extension that your files have in common.

For example, if you wanted to search for all of the files that had the extension .BAK, you would enter

```
SEARCH *.BAK
```

and press **RETURN**.

Likewise, you may also use the wildcard characters in place of a file extension. For example, entering

```
SEARCH REPORT.*
```

and pressing **RETURN** would search for all of the REPORT files, where the file extensions vary.

Error Message

```
Tree structure too deep. Ignoring below this point
```

EXPLANATION: The SEARCH command can only search 32 directory levels deep from the current directory. To alleviate this problem you need to change your current working directory to another, further down in the hierarchical structure, and run the SEARCH command again.

SET (Resident)

Purpose

Sets one string value equivalent to another for use by programs.

Entry Forms

```
SET
SET string1=
```

where ***string1*** is a variable name you wish to delete from the system environment.

```
SET [string1=string2]
```

where ***string1*** is the variable name to which you wish to assign a value; and
string2 is the value you are assigning to *string1*.

Preliminary Concepts

This command is meaningful only if you wish to set values that will be used by various programs or batch files within MS-DOS or that will be used by application programs run under MS-DOS.

SET

An application program can check for all values that have been predefined with this command. The command may also be used to define variables for batch files.

When you define a value with SET, that value remains in the *system environment* until you delete or redefine it, or until you turn off your microcomputer. That is, for a value that has been defined once, you do not need to redefine the value every time you want to start up an application program unless you wish to change the value.

The *system environment* is a series of ASCII strings that must be contained in a limited amount of memory. (Approximately 200 bytes of memory are allocated for the system environment.) The values defined by SET are included in the system environment. The system environment is used by COMMAND.COM and is made available to every program executed under MS-DOS; each program may use some or all of the information provided in the ASCII strings.

The system environment is extremely flexible. Almost anything you wish can be entered in the environment by using the SET command. The only requirement is that correct command line entry form be used when you SET values.

Two variables with predetermined default values are always in the system environment. The values may be changed whenever you wish; however, the variables will revert to their default values whenever you reboot or turn off your system.

The first variable and its default value is PATH= (equivalent to PATH=No Path). The second is COMSPEC=A:\COMMAND.COM. The default value assigned to COMSPEC is the same as the value assigned to SHELL in the system file CONFIG.SYS. (Refer to Chapter 6, "Bootup Features," for more information on CONFIG.SYS.)

Command Line Entry

When using the SET command to define a value, you must enter as *string1* the variable name to which you wish to assign a value. Letters entered for *string1* are automatically converted to uppercase. The assigned value for the variable name is entered as *string2*. The two strings must be separated by an equal sign (=), with no intervening spaces unless a space is part of the string itself.

Note that the characters you enter as part of *string2* are *not* automatically converted to uppercase if they are alpha characters. For this reason, you must be careful to enter the string exactly as it is to be used.

You may define *string1* and *string2* to be anything you want. The only requirement is that you separate these parameters with an equal sign (=) on the SET command line.

If you do not specify a value for *string2* (that is, if you enter **SET *string1*=** and press **RETURN**), variable name *string1* and any currently assigned value will be deleted from the system environment.

Displaying the System Environment

If you wish to display the values that have been defined with SET and are included in the system environment, simply enter **SET** and press **RETURN**. The defined values will be displayed in a list having the format

```
PATH=
COMSPEC=A:\COMMAND.COM
string=string
string=string
.
.
.
string=string
```

NOTE: The values shown above for PATH and COMSPEC are the MS-DOS default values.

Setting Values for Batch Processing

You may use the SET command to define replaceable parameters used in batch files. For example, if you have a batch file containing the statement *LINK %FILE%*, you can use SET to define the file name that MS-DOS will use for that variable, preventing your having to define the variable each time you invoke the batch file during a work session. To set the value of the replaceable parameter *FILE*, you could enter

```
SET FILE=DOMORE
```

and press **RETURN**. Then, whenever *%FILE%* is encountered in a batch file, the system will use the file DOMORE on the default drive.

Deleting Variable Names from the System Environment

The variable names you define using the SET command remain in the system environment until you turn off or reboot your system, preventing your having to reenter them each time you start up an application program. It is also possible to delete variable names and their assigned values from the system environment without rebooting your system. You may do this by entering

```
SET string1=
```

at the system prompt and pressing **RETURN**, where *string1* is the variable name you wish to delete. **NOTE**: Not even a space can follow the equal sign when you are invoking SET to delete a variable. If you enter a space before you press RETURN, the variable will not be deleted from the system environment. Instead, it will be assigned a space as its *string2* value.

For example, suppose you have previously defined the variable name FILE as DOMORE (refer to Setting Values for Batch Processing earlier in this section). Suppose that when you enter **SET** and press **RETURN**, the system environment listing is

```
PATH=
COMSPEC=A:\COMMAND.COM
FILE=DOMORE
```

To delete variable name FILE, you could enter

```
SET FILE=
```

and press **RETURN**. If you displayed the system environment again, you would see that FILE is no longer included.

If you wished to redefine rather than delete the variable name FILE, you would do so by entering

```
SET FILE=NEW DEFINITION
```

and pressing **RETURN**. It is not necessary to delete a variable name before redefining it.

During a given work session, you may wish to delete or redefine variable names in the system environment when you are finished with an application that required certain variables (or values) and need to add some new variables (or values) for another application. Remember that the memory allocated for the system environment is limited.

Error Message

```
Out of Environment Space
```

EXPLANATION: This message is displayed if you try to set a value in the system environment and there is not sufficient memory available to contain the variable name and its value. Approximately 200 bytes of memory are allocated for the system environment.

If this message is displayed, enter **SET** and press **RETURN** to display the current contents of the system environment. If there are some variable names that you no longer need, delete them and then reenter the SET command line.

SHIFT (Resident, Batch-Processing)

Purpose

Under batch file processing, the SHIFT command allows for the use of replaceable parameters past the normal limit of 10.

Entry Form

```
SHIFT
```

Preliminary Concepts

The resident, batch-processing commands are most often executed from within a batch file, although they may be used directly from the command line in some instances. Refer to Chapter 5, "Command Features," for a complete explanation of batch file creation and execution.

NOTE: You must always end each line within a batch file by pressing the RETURN key.

Batch-processing under MS-DOS allows you to specify replaceable parameters. These replaceable parameters (%0 through %9) are used within a batch file as "dummy" parameters which are replaced sequentially with real values when the batch file is executed.

The parameters are substituted in order on the command line. A one-to-one correspondence is set up between the prototype commands in the command line and the replaceable parameters in the batch file. That is, '%1' stands for the first value (name, number, or text) typed after the batch file name, and '%2' stands for the second value typed after the batch file name.

The example command line shown below corresponds on a one-to-one basis with the replaceable parameters directly under it:

batch filename	*filespec1*	*filespec2*	*filespec3* ...	*filespec9*
|	|	|	|	|
%0	%1	%2	%3 ...	%9

NOTE: The replaceable parameter '%0' is always replaced by the drive name (if specified) and the file name of the batch file.

Although you may only you use up to ten replaceable parameters within a batch file (%0 through %9), you can avoid this limitation by using the SHIFT command. The SHIFT command shifts the parameters to allow you more freedom in creating a batch file.

This is accomplished by shifting all parameters one position at a time, to the left. For example, if your batch command was passed, the following parameters:

%0 = "foo"
%1 = "bar"
%2 = "name"
%3...%9 are empty

then a SHIFT will result in the following:

%0 = "bar"
%1 = "name"
%2...%9 are empty

As you can see from the above example, the values (in this case "foo", "bar", and "name") are shifted one position by the use of the SHIFT command.

If there are more than 10 parameters given on a command line, those that appear after the 10th (%9) will be shifted one at a time into the position occupied by %9, in successive shifts.

Command Line Entry

To use the SHIFT command within a batch file, simply type SHIFT. The SHIFT command will shift (to the left) as many parameters as you specify, up to a maximum line length of 128 characters.

For example, you may wish to create a batch file like the one shown below; where the SHIFT command is used, in conjunction with the ERASE command, to delete a number of bad files.

In the following example, a batch file is used to delete a number of bad files by using a single command. This is accomplished by using the SHIFT command to shift the given values over into the replaceable variable, one at a time.

To create a batch file to perform this procedure, enter

```
COPY CON REMOVE.BAT
:LOOP
IF "%1"=="" GOTO DONE
ERASE %1
SHIFT
GOTO LOOP
:DONE
CTRL-Z
```

and press the **RETURN** key. After this batch file has been copied onto your disk, you may use it to erase a number of files with one command. To do this, enter

```
REMOVE filename1 filename2 filename3
```

and press **RETURN**. You should replace the *filename* variables with the names of actual files you want erased.

When you execute the batch file, REMOVE.BAT, you are telling MS-DOS to execute each line of the batch file in order:

1. Execute the statement, `IF "%1"=="" GOTO DONE`

   Now, in our example, the **"%1"** variable will be replaced by the *filename1*. If *filename1* is equal to the string "", which is nothing, enclosed in quotation marks, then the batch file will go to the **DONE** label and the file will terminate.

   However, if *filename1* is not equal to nothing (that is, if it contains data), then the **ERASE %1** command will be executed.

2. Once the ERASE command has been executed, the batch file will SHIFT the next item on the command line, *filename2*, into the position of the replaceable parameter, **%1**.

   When this SHIFT occurs, *filename2* will be erased. This procedure will continue until all files specified in the command line have been erased.

This example of the SHIFT command characterizes its most common usage. Although, it is not necessarily used with the ERASE command but may be used in conjunction with most MS-DOS commands.

SORT (Transient)

Purpose

A filter that reads the standard input, sorts the input in alphabetical or ascending numeric order, then writes the sorted data to the screen display. Commonly used with pipes and input/output redirection.

Entry Form

```
SORT [/x]
```

where */x* is one of the following switches:

/R	Reverse sort (sort from *Z to A* rather than from *A to Z*, or for numeric data, descending rather than ascending order).
/+*n*	Start sort at column *n* (where *n* is an integer value specified by the user).

Preliminary Concepts

The SORT command is a filter that can, through pipes, be used to sort data from other commands or from files in either alphabetical (or reverse alphabetical) or numeric (ascending or descending) order. The sorted data is then output to the screen display (the standard output). You can also use input/output redirection to write the sorted data to another file or to another device.

If no pipe or input/output redirection is used, SORT reads from the standard input (the keyboard) and writes to the standard output (the screen display). For more information on pipes and input/output redirection, refer to Chapter 8, "Input/Output Features."

Command Line Entry

The parameters of the SORT command line are described below.

Command Name

The only required parameter of the SORT command line is the command name. If the command is not located on the default disk, you must precede the command name with the appropriate drive name.

If SORT is used to sort the output of another command, the command line takes the form

```
command | SORT
```

where ***command*** is the command or function whose output you want input to SORT; and
| is the pipe that feeds *command* output to SORT.

Switches

If one or both of the optional switches are used in the command line, they must follow immediately after the command name (SORT). No delimiter other than the switch character (/) that is entered as part of each switch is required, although a space may be inserted for readability if desired. The supported switches are described below.

/R—Reverse Sort

Normally, the SORT command sorts input alphabetically, from *A to Z*, or for numeric data, in ascending order. Use the /R switch to reverse the sort, so that data is sorted and output from *Z to A* or in descending numeric order.

/ + *n*—Start Sort at Column *n*

Normally, the SORT function begins with (sorts on the basis of) the contents of column 1. With the /+*n* switch, you can cause SORT to begin at the *n*th column rather than column 1. The variable *n* is an integer value specified by the user and may be any valid column number.

Sorting the Contents of a File

Suppose you wish to perform a reverse sort on the file UNSORT.TXT that is located on the disk in drive D. Enter

```
SORT /R <D:UNSORT.TXT >B:SORT.TXT
```

and press **RETURN**. SORT will read the contents of the file, sort it (by the first column of each line) in reverse order, then write the output to file B:SORT.TXT. If the destination file for the output of SORT (B:SORT.TXT in this example) does not already exist, it is created.

Sorting Directory Listings

Suppose you wished to have an alphabetical listing of the current directory for the disk in drive B. To obtain a hard copy listing, you could proceed as follows:

- Press **CTRL-PRTSC** to turn on the printer echo of the screen display.
- Enter

  ```
  DIR B: | SORT
  ```

  and press **RETURN**. The sorted directory will be printed as it is displayed on the screen. This effectively provides you an index of the current directory contents.
- Press **CTRL-PRTSC** to turn off the printer echo.

Suppose that instead of an alphabetical listing, you wanted a listing of the files contained in the directory in order of file size. To obtain this, you could enter

```
DIR B: | SORT /+14
```

and press **RETURN**.

If the directory were lengthy and you wished to display the sorted directory listing one screen at a time, you could enter the command

```
DIR B: | SORT /+14 | MORE
```

and press **RETURN**. (Refer to the appropriate section of this chapter for information on the MORE command.)

Error Messages

```
SORT: Incorrect DOS version
```

EXPLANATION: SORT will execute only under MS-DOS version 2 or higher. This message will be displayed if you try to invoke SORT for files that were created under an incompatible version of MS- DOS.

```
SORT: Insufficient disk space
```

EXPLANATION: When used with pipes to sort the output of another command, SORT creates and uses temporary files. This message will be displayed if the disks containing files involved in a SORT operation do not contain sufficient free space for the temporary files.

When this message is displayed, you must move the files you need for the SORT operation to another disk with more space (or erase unneeded files from the disk) and reenter the SORT command line.

```
SORT: Insufficient memory
```

EXPLANATION: This message means that there is not sufficient machine memory available for execution of the SORT operation.

To free up memory so that you can reinvoke the SORT command, you may do one of the following:

- Reboot your system to clear memory of programs that remain resident in memory after execution. (Such programs, called *terminate and remain resident* programs, include the PSC*x* commands that are used to selectively print screen displays.) Reenter the SORT command line.

- If simply rebooting your system does not solve the problem, you can reduce the number of BUFFERS defined in the CONFIG.SYS file. (For information on the CONFIG.SYS file, refer to Chapter 6, "Bootup Features.") After reducing the number of buffers, reboot your system again and reenter the SORT command line.

SYS (Transient)

Purpose

The SYS (Copy System Files) command is used to transfer the MS-DOS system files from the disk in the default drive to the specified drive.

Entry Form

```
SYS d:
```

where ***d*** is the name of the drive to which the system is to be copied.

Preliminary Concepts

SYS is normally used to update the system or to place the system on a formatted disk which contains no files. An entry specifying the destination drive (*d*) for SYS is required.

If IO.SYS and MSDOS.SYS are on the destination disk, they must take up the same amount of space on the disk as the new system will need. This means that you cannot transfer system files from an MS-DOS version 2 (or above) disk to an MS-DOS (Z-DOS) version 1.25 disk. You must reformat the MS-DOS (Z-DOS) version 1.25 disk with the MS-DOS version 2 (or above) FORMAT command before the SYS command can be used.

The destination disk must be completely blank or already contain the system files IO.SYS and MSDOS.SYS.

The files transferred are copied in the following order:

IO.SYS
MSDOS.SYS

IO.SYS and MSDOS.SYS are both hidden files and do not appear when the DIR command is executed. COMMAND.COM (the command processor) is *not* transferred. To transfer COMMAND.COM you must use the COPY command.

Command Line Entry

You need only to enter the SYS command followed by the letter of the destination drive and press RETURN to execute the SYS command.

Example

To place the system on the disk in drive B, enter

```
SYS B:
```

and press **RETURN**. This will cause SYS to display the message:

```
Insert system disk in drive A
and press any key when ready
```

Pressing any key will cause the system files to be placed on the disk in drive B, and SYS will display the following message when finished:

```
system transferred
```

Advanced Concepts

If you are a seasoned programmer, you may eventually use the SYS command to place a new or "patched" IO.SYS on a disk because you need different I/O functions than those that are standard.

Error Messages

```
Incompatible system size
```

EXPLANATION: The system files IO.SYS and MSDOS.SYS do not take up the same amount of space on the destination disk as the new system will need.

```
Incorrect DOS version
```

EXPLANATION: You tried to run a version of SYS that was version 2 or above on a disk with an earlier version of the system on it. The systems must be the same, or the disk must be reformatted.

```
Invalid drive specification
```

EXPLANATION: The SYS command was entered without a drive specification (*d*) or with an invalid drive specification. You must reenter the command.

```
Invalid parameter
```

EXPLANATION: This message will be displayed if you enter anything besides the source drive specification. Reenter the command.

```
No room for system on destination disk
```

EXPLANATION: There is not enough room on the destination disk for the IO.SYS and MSDOS.SYS files. You must reformat the disk with a version of the FORMAT utility 2.0 or above.

```
Not ready error reading drive d
Abort, Retry, Ignore?_
```

EXPLANATION: The drive you designated for SYS does not contain a disk. You must place a disk in the drive.

TIME (Resident)

Purpose

Use to display and/or change (set) the time known and used by the system.

Entry Forms

```
TIME
TIME hh[:mm[:ss[.cc]]]
```

MS-DOS accepts any time you enter as long as the parameters and separators you use are valid. If an invalid parameter value or separator is entered, the time entered is ignored and the system prompts

```
Invalid time
Enter new time:
```

You may then reenter the correct time or press **RETURN** to use the preexisting time.

Command Line Entry

If you invoke the TIME command by entering **TIME *hh*[*:mm*[*:ss*[*.cc*]]]** at the system prompt and pressing RETURN, the system sets its clock to the time you enter without displaying the preexisting time or prompting you to enter the time. After the command is executed, the system prompt is displayed again.

The allowable parameter values for the time you enter are:

hh = a value from 00 to 23, designating the hour;
mm = a value from 00 to 59, designating the minute;
ss = a value from 00 to 59, designating the second; and
cc = a value from 00 to 99, designating the hundredth of a second.

NOTE: The time displayed by the system and accepted with the TIME command is in 24-hour format. Thus, if you wanted to specify 2:00 PM, you would enter 14:00 or 14. Also, note that you are not required to enter values for seconds or hundredths of a second. If you enter a value for the hour and do not specify values for the hundredth of a second, second, or minute parameters, the value of each of those parameters is assumed to be 00.

Only numeric values may be entered; letters are not allowed. The hour, minute, and second parameters must be separated by colons (:). The seconds and hundredths of a second parameters must be separated by a period (.). No other separators are valid.

Usage

Like the DATE command, the TIME command helps you keep track of files for archive purposes. It defines the creation/modification time down into smaller sequences than DATE does. (DATE gives you the date of creation; TIME gives you the time of day the file was created.) This can help you distinguish one file from another if you have worked all day developing a series of files for a program. At the end of the day, you may have several development copies of your files on several disks. The time recorded in the directory for each file, along with the date, gives you a clear reference to the file's currency.

You may also use the TIME command to help you keep track of how long you work on a given project. To measure elapsed time during a work session, you could enter **TIME 00** or **TIME 0** at the beginning of the session to set the time at "0." Then, at the end of your session, you could simply enter TIME with no parameters, and the time displayed would be the elapsed time.

Error Message

```
Invalid time
Enter new time:
```

EXPLANATION: You have entered an invalid time. You may have entered too much or too little information, an invalid parameter value, or used invalid separators. Reenter the time.

TREE (Transient)

Purpose

Displays all directory paths on the default or specified disk and, optionally, lists the files contained in each subdirectory.

Entry Forms

```
TREE ?
TREE [d:][/F]
```

where ? invokes the TREE help screen display,
d: is the drive name identifying the disk (if other than the default) for which you wish to display the subdirectory structure, and
/F is the optional Files switch that causes TREE to list the files contained in each subdirectory.

Preliminary Concepts

The TREE command enables you to display the path names for all the subdirectories that exist on the default or specified disk. By using the /F (Files) switch, you can also display the file names of the files that are contained in each subdirectory. Thus, the TREE command is useful when you wish to review the directory structure of a particular disk or when you wish to review the contents of a disk that includes one or more subdirectories.

Note that the TREE command will not display the root directory (\) as a path or the names of files in the root directory. Use DIR or DIR \ to display the contents of the root. (Refer to the appropriate section of this chapter for information about the DIR command. For information about the MS-DOS directory structure, refer to Chapter 7, "Directory Features.")

If the default or specified disk has no subdirectories (if all disk contents are in the root directory), TREE will display the message:

```
No sub-directories exist
```

followed by the system prompt.

For disks that *do* include subdirectories, TREE produces screen displays in the form:

```
DIRECTORY PATH LISTING FOR VOLUME label


Path:  pathname

Sub-directories:  dirname
                     .
                     .
                     .
                  dirname

Files:            filename
                     .
                     .
                     .
                  filename
```

where *label* is the disk volume label,
pathname is a directory path name identifying a unique subdirectory,
dirname is the name of a subdirectory, and
filename is the primary file name and extension (if one exists) of a unique file.

The TREE display is followed by the system prompt.

If a disk was not assigned a volume label when it was formatted, the *label* in the display header will be shown as ???????????.

A `Path:` entry is displayed for every subdirectory found, along with a `Sub-directories:` entry. If a given path (subdirectory) has no subdirectories, then the entry shows `Sub-directories: None`.

When (and only when) you use the /F (Files) switch, a `Files:` entry is also shown for every subdirectory (`Path:`) that TREE finds. If a subdirectory contains no files, then this entry shows `Files: None`.

You may use input/output redirection with the TREE command to obtain a printed listing of TREE output or to send output to a file. You may also wish to use TREE in conjunction with the MORE command to make the screen display easier to read. If a PSC utility is used in this case, you may selectively print portions of TREE output.

Input/output redirection and printing screen displays are described in Chapter 8, "Input/Output Features." The MORE command and PSC utilities are described elsewhere in this chapter.

TREE Help Screen

To display the TREE help screen, enter:

```
TREE ?
```

at the system prompt and press **RETURN**. The screen will display a summary of TREE usage and valid command line entry forms as shown in Figure 11.20a. The help screen is followed by the system prompt, making it easy for you to refer to the information provided as you enter a TREE command line.

```
                    TREE Version 2.xx


TREE  shows  the subdirectory structure of the default  or  specified
disk.  If the optional /F (Files) switch is used, TREE also lists the
files contained in each subdirectory.

Syntax: TREE ?
        TREE [d:][/F]
```

Figure 11.20a. TREE Help Screen

Command Line Entry

To invoke TREE for the default disk, enter:

```
TREE
```

at the system prompt and press **RETURN**. (If TREE is not on the default disk, you must precede the command name with the appropriate drive name (*d*:).)

If you wish to display the subdirectories of a nondefault disk, enter:

```
TREE d:
```

at the system prompt (where *d*: is the drive name identifying the desired disk) and press **RETURN**.

If you wish to display lists of the files contained in each subdirectory as well as the path name and the subdirectories for each, then enter the /F (Files) switch as part of the TREE command line. This switch may be entered either before or after the drive name parameter (if used) and may or may not be separated from it by a space or equivalent MS-DOS delimiter. The switch must, however, be entered *following* the command name (TREE).

Thus, if you use the /F switch when you invoke TREE for the default disk, you may enter a command line in the form:

```
TREE/F
```

or

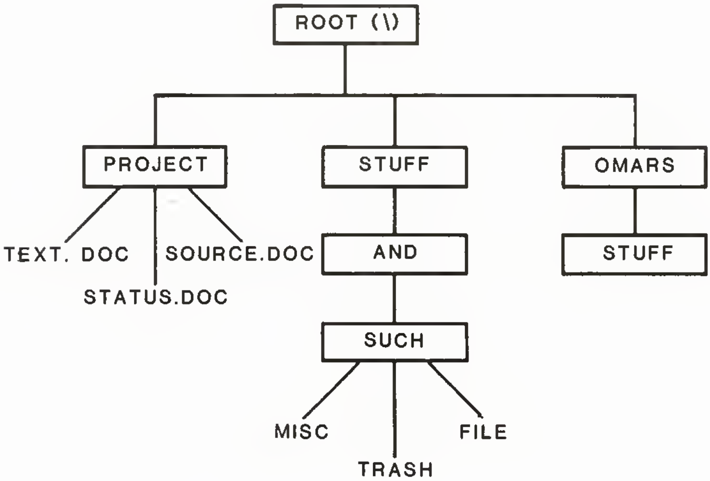
```
TREE /F
```

When you use the /F switch when you invoke TREE for a nondefault disk, you may enter a command line in any of the following forms:

```
TREE d:/F
TREE d: /F
TREE/Fd:
TREE /Fd:
TREE /F d:
```

Examples

Suppose you have a disk named STUFF in drive C and that the following illustrates the directory structure and contents of the disk:



To display all subdirectories on this disk and the files contained in each, you could enter:

```
TREE C:/F
```

at the system prompt and press **RETURN**. The screen would display:

```
DIRECTORY PATH LISTING FOR VOLUME STUFF


Path:  \PROJECT

Sub-directories:  None
```

```
Files:            TEXT    .DOC
                  SOURCE  .DOC
                  STATUS  .DOC


Path:  \STUFF

Sub-directories:  AND

Files:            None


Path:  \STUFF\AND

Sub-directories:  SUCH

Files:            None


Path:  \STUFF\AND\SUCH

Sub-directories:  None

Files:            MISC
                  TRASH
                  FILE
Path:  \OMAR'S

Sub-directories:  STUFF

Files:            None


Path:  \OMAR'S\STUFF

Sub-directories:  None

Files:            None
```

If you invoked TREE without the /F switch, then the entries for the files contained in each subdirectory would not be displayed.

If you wished to save the output of TREE to a file in the above example, you could have entered:

```
TREE C:/F >TREE.FIL
```

at the system prompt. The information in the above example display would have been written to a file named TREE.FIL on the default disk.

You can also use input/output redirection to obtain a hard copy of the directory structure of a disk. For the above example disk, you could enter:

```
TREE C:/F >PRN
```

at the system prompt and press **RETURN**.

Error Messages

```
Incorrect DOS version
```

EXPLANATION: This message will be displayed if you booted up your system with a version of MS-DOS previous to version 2. To use TREE, MS-DOS version 2 must be resident in memory.

```
Invalid drive specification
```

EXPLANATION: This message is displayed if you entered an invalid drive name as part of the TREE command line. You may have entered a drive name that is not supported by MS-DOS, or you may have entered a drive name that does not exist in your system. Reenter the command line, making sure that you specify a valid drive name.

```
Invalid parameter
```

EXPLANATION: This message is displayed if you entered a TREE command line with a syntax error or one that included an invalid parameter. Reenter a valid TREE command line.

TYPE (Resident)

Purpose

Displays the specified file's contents on the screen without altering the file.

Entry Forms

```
TYPE filespec
TYPE [d:][pathname]filename
```

where ***filespec*** identifies the file you wish to display,
d is the drive name identifying the disk (if other than the default) on which the file is located,
pathname identifies the directory (if other than the current directory of the default or specified disk) in which the file is located, and
filename is the primary file name (and extension, if any) of the file you wish to display.

Preliminary Concepts

When you invoke the TYPE command, MS-DOS displays the contents of the specified file on the screen. The contents of the file are copied from the disk to the CRT. TYPE does not alter the contents of the displayed file. If you wish to change file content, you must use a text editor program or EDLIN (see Chapter 12 of this manual).

TYPE copies file contents to the screen in the same fashion as COPY does when you use it to copy a file to a device. That is, for any given file, entering either

```
TYPE filespec
```

or

```
COPY filespec CON
```

at the system prompt and pressing **RETURN** will produce the same results. In the COPY command line, **CON** is the device name for the screen display; TYPE automatically sends file contents to the screen display without your having to specify the device, CON.

If you use TYPE to display binary files, all control characters (including bells, form feeds, and escape sequences) will be sent to your microcomputer.

The only formatting the system performs on the file's screen display is that any tab stops are expanded to every eighth blank column, as is consistent with having tab stops every eight columns.

NOTE: When using TYPE to display files, you may find it helpful to also use the MORE command to display the file one screen at a time. (Refer to MORE in this chapter for additional information.) As an alternative, you may press **CTRL-NUMLCK** to temporarily stop and restart the display.

Command Line Entry

The parameters of the TYPE command line are described in this part. You must always specify the file to be displayed, but the parameters you must input as part of the file specification depend upon the file's location with respect to the default disk and current directories.

File Specification

You must enter a *filespec* following the TYPE command to specify the file you wish to have displayed on the screen. If the file is in the current directory of the default disk, you need only enter the primary file name (and extension, if any) of the file.

Drive Name

If the file you wish to display is on a disk other than the default disk, you must enter the drive name (*d*) identifying the drive in which the disk is located. If no drive name is entered, the system will search the default drive for the file you specify.

Path Name

If the file you wish to display is not in the current working directory of the default or specified disk, you must enter a valid *pathname* for the system to locate the proper directory.

Using TYPE to Display a File on the Default Disk

Suppose you have a reference file named PHONE.LST in the current directory of the default disk and wish to display the file to find a particular telephone number contained in the file. You may do this by entering

```
TYPE PHONE.LST
```

and pressing **RETURN**. The contents of the file will be displayed on your screen.

If the file is on the default disk but not in the current working directory, you must enter the path name for the system to find the correct directory. For example, if the file PHONE.LST were in a second-level directory named CONTACTS, you would have to enter the following to display the file on your screen:

```
TYPE \CONTACTS\PHONE.LST
```

Using TYPE to Display a File on a Non-Default Disk

If you wish to display the contents of a file on a disk other than that in the default drive, you must enter a valid drive name as part of the command line. If the file is not in the current directory of the disk, you must also enter the path name for the system to find the directory in which the file is located. For example, suppose you wish to display file TEXTFILE that is in subdirectory PRACTICE on the disk in drive F. You could display the file's contents by entering

```
TYPE F:\PRACTICE\TEXTFILE
```

and pressing **RETURN**.

Error Messages

```
File not found
```

EXPLANATION: You have entered an invalid file specification or path name, or the file you specified is not on the default or specified disk. Reenter the TYPE command, making sure the file specification is correct.

```
Invalid number of parameters
```

EXPLANATION: You have entered the TYPE command without the required file specification. Reenter the command, specifying the file that you wish to be displayed.

```
Not ready error reading drive d
Abort, Retry, Ignore?
```

EXPLANATION: When you entered the TYPE command, you specified a drive name of a drive that does not contain a disk, a drive with a door that is not properly closed, or a drive that does not exist or is not configured for your system. If the drive does not exist or is not configured for your system, enter **A** to abort the attempted operation. The system prompt will be displayed. You may then reenter the TYPE command using a valid drive name.

If the drive does not contain a disk or if the drive door is open, insert a disk and/or close the drive door and then enter **R**. The system will reattempt the TYPE operation.

VER (Resident)

Purpose

The VER (Version) command displays the version number of MS-DOS and IO.SYS on your screen.

Entry Form

```
VER
```

Command Line Entry

If you want to know what version of MS-DOS you are currently using, type

```
VER
```

and press the **RETURN** key. The version number will be displayed on your screen. The MS-DOS version number is composed of a single-digit number (the major version level) followed by a decimal point and a double-digit number (the minor revision level).

For example, if you typed

```
VER
```

and pressed **RETURN**, your screen would display:

```
IO.SYS Version 2.00
MS-DOS Version 2.00
```

NOTE: The version number that appears on your screen may differ from that shown in the example.

VERIFY (Resident)

Purpose

The VERIFY command is used to verify that data has been correctly written to disk.

Entry Form

```
VERIFY [ON]
VERIFY [OFF]
```

where **ON** turns on the VERIFY utility; and
OFF turns off the VERIFY utility.

Preliminary Concepts

This command causes the same *type* of verification procedure to be executed as the /V switch does when the COPY command is used. The major difference is that once VERIFY has been turned on, it stays on until it is turned off. If you want to verify that all files have been written to disk correctly, you can use the VERIFY command to instruct MS-DOS to verify that your files are intact (no bad sectors, for example). When VERIFY is ON, MS-DOS will perform a VERIFY *every* time you write data to a disk. Because VERIFY requires extra time to perform these verifications, your system will run somewhat slower when writing data to disk.

You will receive an error message only if MS-DOS was unable to successfully write your data to disk.

VERIFY ON remains in effect until you issue a VERIFY OFF command to MS-DOS.

If you want to know the current setting of VERIFY, type VERIFY with no options. The screen will display the current state (*on* or *off*) of the verify option.

Command Line Entry

```
VERIFY ON
```

Turns VERIFY on.

```
VERIFY OFF
```

Turns VERIFY off.

If you want to know the current setting of VERIFY, enter

```
VERIFY
```

and press the **RETURN** key. The screen will display:

```
VERIFY is on
```

if VERIFY is on, or

```
VERIFY is off
```

if it is off.

Advanced Concepts

VERIFY ON remains in effect until you change it. You may change the status of VERIFY by including a SET VERIFY system call within a program.

VOL (Resident)

Purpose

The VOL (Volume) command displays the disk volume label, if one exists.

Entry Form

```
VOL [d:]
```

Preliminary Concepts

This command displays the volume label of the disk in the designated drive. (A disk volume label is a label of up to eleven characters, which may be entered at the time a disk is formatted.) If no drive is specified, MS-DOS prints the volume label of the disk in the default drive.

If the disk does not have a volume label, VOL displays:

```
Volume in drive d has no label
```

Command Line Entry

If you enter:

```
VOL
```

and press **RETURN**, your screen will display:

```
Volume in drive d is volume label
```

where *volume label* is the eleven character disk label of the specified disk.

VOL

NOTE: Disk volume labels will always be displayed in uppercase.

If the disk being checked has no volume label (none was entered when the disk was formatted), the following message will display:

```
A>Volume in drive d has no label
```

Part IV

Program Development Command Guide

Program Development Command Guide

This guide provides a comprehensive description of four of the commands in your MS-DOS software package that can be beneficial to users who wish to develop software programs.

Knowledge of these commands is not essential for most users who do not wish to develop software programs.

These commands are explained in terms that can be understood by users who have previous experience in microcomputer programming.

Furthermore, it is neccessary that you understand the concepts and perform the necessary procedures in Part I, "Preparation Guide" of this manual. Familiarity with some sections of Part II, "Primary Feature Guide" is also helpful for using these commands.

The sequence of the chapters in this guide corresponds to the sequence in which these commands will probably be used. This guide consists of the following four chapters:

Chapter 12, "EDLIN"—This chapter includes explanations of EDLIN's purpose, entry form, preliminary concepts, usage examples, and error messages. EDLIN is a line editor that enables you to create, edit, and display source program and text files.

Chapter 13, "LIB"—This chapter includes explanations of LIB's purpose, entry forms, preliminary concepts, usage examples, and error messages. LIB deletes modules from a library file, changes library modules into object files, and appends object files to a library file.

Chapter 14, "LINK"—This chapter includes explanations of LINK's purpose, entry forms, preliminary concepts, usage examples, and error messages. LINK combines several object modules into one relocatable load module, or run file.

Chapter 15, "DEBUG"—This chapter includes explanations of DEBUG's purpose, entry form, preliminary concepts, usage examples, and error messages. DEBUG provides a controlled testing environment for isolating and eliminating errors in, or malfunctions of, binary and executable program files. It enables you to alter the contents of a file or the contents of a CPU register and then to reexecute a program immediately to check the changes.

NOTE: The EXE2BIN command (described in Chapter 11, "Command Descriptions") can also be used in program development.

If you need information about commands for preparing a Winchester disk, refer to Part V, "Winchester Command Guide."

If you need information about commands that perform more basic functions, refer to Part III, "Primary Command Guide."

Chapter 12
EDLIN

Overview

EDLIN is an MS-DOS utility that enables you to create, edit, and display source program and text files. EDLIN is called a *line editor* because the text in files created or edited by EDLIN is divided into lines, each of which may be up to 253 characters in length. While you may use system editing and control keys to make changes within a given line, the EDLIN program commands are operative upon whole lines or specified groups of lines.

Command Entry Form

```
EDLIN filespec
```

where ***filespec*** is the file specification of the file you wish to edit.

Preliminary Concepts

You can use EDLIN to:

- create and save new source files or text files.
- update existing files and save both the updated and original versions.
- edit, delete, insert, and display lines.
- search for, delete, and replace text strings within one or more lines within a file.

Since EDLIN enables you to create and manipulate file contents on the basis of lines of text or data, dynamic line numbers are generated and displayed by EDLIN but are not actually saved as part of the file. When a file is created or edited, the line numbers begin at 1 and are increased by one for each line through the end of the file.

When new lines are inserted between existing lines, all the line numbers following the inserted lines are automatically increased by the number of lines inserted. When lines are deleted from an existing group of lines in a file, all line numbers following the deleted lines are automatically decreased by the number of lines deleted. Thus, the lines within any file are always numbered consecutively.

EDLIN is invoked by inputting an EDLIN command line at the system prompt. Once EDLIN is loaded into memory, there are two types of commands or functions that you can use in creating and editing files: EDLIN program commands, and MS-DOS editing and control key functions. EDLIN program commands are used to manipulate whole lines of text or data and are called *interline commands*. MS-DOS editing and control keys are used to edit the text or data within one line and are known as *intraline editing functions*, as applied to EDLIN. (All MS-DOS editing and control keys are described from a general system viewpoint in Chapter 5, "Command Features.")

Invoking EDLIN

To invoke EDLIN you must enter at the system prompt a command in the form

```
EDLIN filespec
```

where ***filespec*** is the full file specification of the file you wish to create or edit.

If the file is in (or is to be in) the current directory of the default disk, you may simply enter the primary file name and extension. If the file is on another disk or in another directory, you must include the appropriate drive name and/or path name as part of the file specification. You may not use wildcard characters in the EDLIN *filespec*.

Invoking EDLIN

When you press RETURN, EDLIN is loaded into memory and searches for the specified file. If the file you specify does not already exist (that is, if a file matching the specification you entered is not found), EDLIN will display the message

```
New File
```

followed by the EDLIN prompt (*). You may then begin file creation.

If the file you specify when you invoke EDLIN does exist, the file is loaded into working memory. If the entire file is loaded, EDLIN will display the message

```
End of input file
```

followed by the EDLIN prompt (*). If the file is too large to be contained in available working memory, EDLIN will load as much of the file as possible and then display the EDLIN prompt (*). The absence of the `End of input file` message lets you know that the entire file was not read into memory. The EDLIN program APPEND LINES and WRITE LINES commands enable you to sequentially read and write blocks of a file that is too large to be loaded into available working memory at one time.

When the EDLIN prompt (*) is displayed, you may begin editing the file by using the interline commands and intraline editing functions.

When you are finished with the editing session, you can either use the QUIT EDIT command to abandon edit or the END EDIT command to save the edited (new) and original files on disk. If you use the QUIT EDIT command, any lines you created or changed will not be saved to disk. (If you entered the specification for a new file when you invoked EDLIN, the file will not be created on disk when you use the QUIT EDIT command.)

If you use the END EDIT command, the new or edited file is written to disk using the file specification you entered when you invoked EDLIN, and the original file (if any) is renamed to have a .BAK extension. Thus, both the original and edited versions of the file are saved on disk. If a .BAK file already exists for a file you edit, that preexisting backup file will not be erased until the end of the editing session or until the disk space it occupies is required by EDLIN.

You cannot use EDLIN to edit any file with the extension .BAK because EDLIN assumes that any file with this extension is a backup file. If you find it necessary to edit a .BAK file, you must first rename the file or copy the file to a destination file with a different extension. Then you may invoke EDLIN using the new file name in the command line file specification. (See RENAME and COPY in Chapter 11, "Command Descriptions.") Copying the file is recommended so that you retain a backup file.

Interline Commands

EDLIN interline commands are entered at the EDLIN prompt (*) to edit, insert, or delete one or more specified lines. The interline commands and their entry forms are listed in Table 12.1.

As shown in Table 12.1, most interline commands accept one or more optional, user-defined parameters. While the same parameter may be entered for more than one command, the effect of the parameter depends upon the command with which it is used. A general description of the parameters and their possible values is provided below. Detailed descriptions of the commands and specific results of the parameters used with them are provided following Interline Command Parameters.

Table 12.1. EDLIN Interline Commands

COMMAND	ENTRY FORM
APPEND LINES	`[n]A`
DELETE LINES	`[line1][,line2]D`
EDIT LINE	`line`
END EDIT	`E`
INSERT LINE	`lineI`
LIST LINES	`[line1][,line2]L`
QUIT EDIT	`Q`
REPLACE TEXT	`[line1][,line2][?]R[string1]F6[string2]`
SEARCH LINES	`[line1][,line2][?]S[string]`
WRITE LINES	`[n]W`

Interline Command Parameters

The *line* parameter indicates a user-specified line number. In actual command entry, the *line* may be specified in one of three ways:

- An integer value from the range of 1 to 65533, inclusive. If you specify a line number greater than the highest existing line number within a file, then EDLIN assumes that you mean the line following the last assigned line number. For example, if you are editing a file that contains 256 lines and you enter an interline command that includes line number 458, EDLIN will ignore the actual line number value you entered and assume line number 257.

- Period (.). Use a period (.) to indicate the current line number, where the current line is the last line edited. The last line edited is not necessarily the last line displayed. The current line is always identified by an asterisk (*) between the line number and the first character of the line itself.

- Pound (#). Use the pound sign (#) to indicate the line after the last assigned line number in the file. Specifying the pound sign for *line* has the same effect as specifying a number larger than the last line number.

In the SEARCH LINES or REPLACE TEXT commands, the question mark (?) parameter directs EDLIN to query you whether the correct string has been found. This parameter is used only with the SEARCH LINES and REPLACE TEXT commands.

The *string* parameter indicates a user-specified text string for EDLIN to find, to replace, or to insert in the place of other text. Like the question mark, a *string* is used only with the SEARCH LINES and REPLACE TEXT commands. Terminate each string in a given command line by pressing F6 or RETURN. No spaces should be left between strings or between a string and its command letter unless spaces are a part of the string itself.

The *n* parameter represents an integer value from the range of 1 to 65533, inclusive. This component is used with the APPEND LINES and the WRITE LINES commands to specify the number of lines to be appended to memory or written to disk.

APPEND LINES Command

The APPEND LINES command is used to read lines from the input file (that is, the file being edited) and write them to the editing input buffer. The command is entered at the EDLIN prompt in the form

```
[n]A
```

where *n* is the number of lines you wish to have read into the buffer.

Use this command when you are editing a large file that will not fit into working memory (the edit buffer) all at one time. When EDLIN is invoked and the file matching the input file specification is found, as much of the file as possible is read into the edit buffer. If the file is too large to be contained within the buffer, EDLIN will load as much of the file as possible and you may begin editing the first portion of the file. When you are ready to edit the next block of the file's contents, you must first write the edited buffer contents to the disk using the WRITE LINES command. After freeing buffer space by writing out to the file, you may then append additional lines from the file to the edit buffer with the APPEND LINES command.

The *n* parameter enables you to specify the number of lines you want EDLIN to read into the buffer. If you do not specify a value for *n* (that is, if you simply enter A at the EDLIN prompt and press **RETURN**), then lines are appended to the current contents of the buffer either until the buffer is 3/4 full or until there are no more lines to be appended to the buffer. If the remainder of the file is read into the buffer during an APPEND LINES operation, the screen displays the `End of input file` message followed by the EDLIN prompt. If only part of the file is read into the buffer, the EDLIN prompt is displayed.

DELETE LINES Command

The DELETE LINES command is used to delete one or more specified lines from the file being edited. The command is entered at the EDLIN prompt in the form

```
[line1][,line2]D
```

where ***line1*** is the line number of the first line in a block of text to be deleted, and
line2 is the line number of the last line in the block of text to be deleted.

Line1 and line2 may be specified as: integers, the pound sign (#), or a period (.) as described under Interline Command Parameters. In any case, the two parameters must be separated by a comma (,).

If no value is entered for line1 (in other words, if you enter ***,line2*D** or ***line2*D** and press **RETURN**), the default value for line1 is the current line. A command in this form causes EDLIN to delete the current line, the specified line, and any lines in between those two lines.

If no value is entered for line2, the value for line2 defaults to that specified for line1. Thus, if you wish to delete only one line rather than multiple lines, enter the DELETE LINES command in the form

```
line1,D
```
or
```
line1D
```

Whenever you delete a line or lines, the remaining lines in the file are automatically renumbered so that all lines are still numbered consecutively. The line immediately following the deleted line or group of lines becomes the current line and is assigned the line number that was associated with the (first) line deleted.

Deleting Multiple Lines

Assume that the following file exists and is ready to edit:

```
  1: This is a sample file.
  2: Use: to demonstrate dynamic line numbers
  3: See what happens when you
  4: Delete and Insert
   .
   .
   .
 25: (The D and I commands)
 26: (Use CTRL-BREAK to exit the EDLIN interline insert mode)
 27:*Line numbers
```

Now suppose that you want to delete lines 5 through 24 from the file. To do this, you must enter the following command at the EDLIN prompt and press **RETURN**:

```
5,24 D
```

After completion of the delete operation, the file appears as follows if displayed with the LIST LINES command:

```
1: This is a sample file.
2: Use: to demonstrate dynamic line numbers
3: See what happens when you
4: Delete and Insert
5:*(The D and I commands)
6: (Use CTRL-BREAK to exit the EDLIN interline insert mode)
7: Line numbers
```

Notice that the current line is now line 5, as indicated by the asterisk (*) between the line number and the first character of the line itself.

NOTE: You may use the DELETE LINES command to delete multiple lines only if the lines are in sequence, as in the example above. If you wish to delete more than one line but the lines are *not* adjacent to each other, you must delete each line individually, as described under Deleting a Single Line.

Deleting a Single Line

In the sample file shown at the end of Deleting Multiple Lines above, suppose you wanted to continue your editing session by deleting a single line, line 6. To do this, you must enter

```
6D
```
or
```
6,D
```

at the EDLIN prompt and press **RETURN**. When the line has been deleted, the file appears as follows if displayed with the LIST LINES command:

```
1: This is a sample file.
2: Use: to demonstrate dynamic line numbers
3: See what happens when you
4: Delete and Insert
5: (The D and I commands)
6:*Line numbers
```

Deleting a Range of Lines that Includes the Current Line

Suppose you have the following file loaded and ready to edit:

```
1:  This is a sample file.
2:  Use: to demonstrate dynamic line numbers
3:*See what happens when you
4:  Delete and Insert
5:  (The D and I commands)
6:  (Use CTRL-BREAK to exit the EDLIN interline insert mode)
7:  Line numbers
```

If you wish to delete lines 3 through 6, you can do so by entering

```
,6D
```

and pressing **RETURN**. The default value for the *line1* parameter is the current line. Since the current line (as shown by the asterisk) is 3, EDLIN will delete lines 3 through 6. When the delete operation is complete, the file appears as follows if displayed with the LIST LINES command:

```
1: This is a sample file.
2: Use: to demonstrate dynamic line numbers
3:*Line numbers
```

EDIT LINE Command

The EDIT LINE command enables you to load any line into the edit buffer for editing. The line specified as part of an EDIT LINE command becomes the current line, as indicated by the asterisk displayed between the line number and the first character of the line itself. The EDIT LINE command is entered at the EDLIN prompt (*) in the form

```
line
```

where ***line*** is the number of the line you wish to edit.

The *line* may be specified as an integer value or the pound sign (#) as described under Interline Command Parameters. If you simply press **RETURN** at the EDLIN prompt without having entered a line number, the line after the current line becomes the new current line.

For example, suppose you have the following file listed on your screen

```
1: This is a sample file.
2: Use: to demonstrate dynamic line numbers
3: See what happens when you
4:*Delete and Insert
5: (The D and I commands)
6: (Use CTRL-BREAK to exit the EDLIN interline insert mode)
7: Line numbers
```

Notice that the current line is line 4. If you press **RETURN** at the EDLIN prompt, the current line becomes line 5. You may then press RETURN to leave the line unchanged or edit it as desired. If you wished to make line 7 the current line for editing, you could enter **7** at the EDLIN prompt and press **RETURN**.

When you specify a line for editing, the line number and the line itself are displayed. The line number, followed by the cursor, is displayed immediately below the line. In the case of the last example above, the display would appear as

```
7: *Line numbers
7: *
```

The displayed line is moved to the template when you enter the EDIT LINE command. This means that you can use any of the intraline editing functions to edit the line as desired. (Refer to Intraline Editing Functions in this chapter for information on the template and how it is used.) If you do *not* want to change the line and the cursor is positioned at the beginning or the end of the line, you may simply press **RETURN** and the line will be unchanged. The line following the unchanged line will become the current line.

NOTE: If the RETURN key is pressed while the cursor is in the middle of a line, everything contained in the line after the cursor position will be truncated or deleted.

END EDIT Command

Use the END EDIT command to exit the EDLIN program and save both your edited and original (if file was read from disk for editing and is not a newly-created file) files to disk. When the file or files have been saved, the system prompt will be displayed.

To invoke this command, enter

```
E
```

at the EDLIN prompt and press **RETURN**. The edited file will be written to disk according to the file specification you entered when you invoked EDLIN. The original file (if you were editing a file that already existed) will be renamed to have the original primary file name and the extension .BAK to indicate that it is a backup file. Backup files are created by EDLIN only when you have been editing a preexisting file, not when you first create a file.

The END EDIT command cannot be entered with any parameters; it can only be invoked by entering **E** and pressing **RETURN**. Thus, at the end of an editing session, it is not possible for you to direct EDLIN to write the files to a disk in a specific drive. If you want a file to be written to a disk other than the default disk (and/or directory other than the current directory), you must enter the appropriate drive name (and/or directory path name) when you enter the EDLIN command line at the beginning of the session. If no drive name was entered in the EDLIN command line, the file will be written to the disk in the default drive when you invoke the END EDIT command. If no directory path name was entered when you invoked EDLIN, the file will be written to the current directory.

The above points are especially important to remember when you are creating files. It is always possible to COPY a file from one disk to another, but specifying a drive name as part of the file specification you enter when you invoke EDLIN may save you time and trouble. You should also be certain that the disk you specify contains enough free space to accommodate the file you are creating; or, if you are editing an existing file, the disk should contain enough free space to accommodate both the original and the edited file. If a disk does not have enough free space when you invoke the END EDIT command, the saving of the file(s) is aborted and at least part of the edited file will be lost.

INSERT LINE Command

The INSERT LINE command enables you to insert one or more lines into a file (whether you are creating a file or editing an existing file) by turning on the EDLIN interline insert mode. When you are creating a file and have invoked EDLIN, you must enter the INSERT LINE command before you start entering information into the file.

NOTE: The EDLIN insert mode invoked with the INSERT LINE command is not to be confused with the intraline insert mode described under Intraline Editing Functions. The EDLIN insert mode enables you to add a line or group of lines to a file. The intraline insert mode enables you to insert characters in a given line without changing or deleting characters already in the template.

The INSERT LINE command is entered at the EDLIN prompt in the form

```
lineI
```

where ***line*** is the line number of the line immediately before which the insertion is to be made.

The *line* may be specified as an integer value or the pound sign (#) as described under Interline Command Parameters. If no value is specified for *line* (in other words, if you enter I and press **RETURN**), the default line is the current line.

Once you turn on the insert mode using the INSERT LINE command, EDLIN stays in the insert mode until you enter **F6 RETURN, CTRL-Z RETURN,** or **CTRL-BREAK**. This enables you to insert more than one line, since lines within a file are terminated simply by a **RETURN**.

As you insert lines, the line numbers within the file are automatically increased by one for each line inserted. When you turn off the insert mode with an F6 RETURN or a CTRL-BREAK, the current line is the line immediately following the inserted lines. (That is, the current line is the text or data line you specified in the INSERT LINE command. Note, however, that the line number will be different since EDLIN automatically renumbers existing lines to accommodate inserted or deleted lines.)

Inserting Text before a Specified Line

Assume that the following file has been loaded with EDLIN and is ready for editing:

```
1: This is a sample file.
2: Use: to demonstrate dynamic line numbers
3: See what happens when you
4: Delete and Insert
5: (The D and I commands)
6: (Use CTRL-BREAK to exit the EDLIN interline insert mode)
7:*Line numbers
```

To insert two lines before line 4, enter

```
4I
```

and press **RETURN**. The screen displays

```
4:*_
```

You may then enter the two lines as follows:

- Type **fool around with** and press **RETURN**.
- Type **those very useful commands that** and press **RETURN**.
- Press **F6 RETURN** or **CTRL-BREAK** to end the EDLIN interline insert mode.

  If you were then to display the file (refer to LIST LINES Command), it would appear as follows:

```
1: This is a sample file.
2: Use: to demonstrate dynamic line numbers
3: See what happens when you
4: fool around with
5: those very useful commands that
6:*Delete and Insert
7:  (The D and I commands)
8:  (Use CTRL-BREAK to exit the EDLIN interline insert mode)
9: Line numbers
```

Inserting Lines Immediately before the Current Line

Assume that you want to add another line to the above sample file immediately before the current line (6). To do this, simply enter **I** and press **RETURN**. Since the default value for the *line* parameter is the current line, the screen displays `6:*` to show that this line is ready for you to insert text. You may then type in

```
perform the two major editing functions,
```

and press **RETURN**. Then, to end the insert mode, press **CTRL-BREAK** or **F6 RETURN**.

NOTE: You must press RETURN to mark the end of a line that you insert *before* you end the insert mode with CTRL-BREAK or F6 RETURN. If you exit the insert mode by pressing CTRL-BREAK before you have pressed RETURN at the end of a line, the line you entered will not be added to the file. If you try to exit the insert mode by pressing F6 RETURN before you

NOTE: You may list any range of lines that you wish. If, however, the number of lines in the range you specify is greater than the screen capacity, the display will scroll too quickly to read unless you use the CTRL-NUMLCK key as a toggle to stop and restart it.

If no value is entered for line1 (that is, if you enter ***,line2*L** and press **RETURN**), the default value for the first line is the line number 11 lines before the current line. That is, EDLIN will display the range of lines that begins 11 lines before the current line and ends at the specified *line2*.

If no value is entered for line2 (that is, if you enter either *line1*,L or *line1*L and press **RETURN**), the default value for the second line is the line number 22 lines after the specified line. That is, EDLIN will display the specified *line1* and the 22 lines immediately following that line.

If no values are entered for either line1 or line2 (that is, if you enter **L** and press **RETURN**), EDLIN displays the current line, the 11 lines preceding the current line, and the 11 lines following the current line.

Whenever you display a group of lines that includes the current line, the current line is identified by an asterisk (*) between the line number and the first character of the line itself.

Listing a Specified Group of Lines

Assume that the following file exists and is ready for you to edit:

```
1: This is a sample file.
2: Use: to demonstrate dynamic line numbers
3: See what happens when you
4: Delete and Insert
5: (The D and I commands)
.
.
.
```

```
15:*The current line contains an asterisk.
 .
 .
 .
26: (Use CTRL-BREAK to exit the EDLIN interline insert mode)
27: Line numbers
```

To display a range of lines without reference to the current line, you must enter a LIST LINES command with both line parameters specified. For example, to display lines 2 through 5, inclusive, enter

```
2,5L
```

and press **RETURN**. The screen displays lines 2 through 5.

Listing a Group of Lines with Reference to the Current Line

In the last sample file listed, notice that the current line number is 15. To display a range of lines that begins 11 lines before the current line and ends at a specified line, you could enter

```
,26L
```

and press **RETURN**. Since the line 11 lines before the current line is line 4 and you specified line 26 as the last line of the range, the screen displays

```
 4: Delete and Insert
 .
 .
 .
15:*The current line contains an asterisk.
 .
 .
 .
26: (Use CTRL-BREAK to exit the EDLIN interline insert mode)
```

Listing a Group of Lines with Reference to a Specified Line

If you wish to display a group of lines that begins with a specific line and ends at the twenty-second line after the specified line, you may enter a LIST LINES command that includes a value only for the first line to be displayed. For example, if you wanted to display a section of a file beginning with line number 13, you could enter

```
13,L
```
or
```
13L
```

and press **RETURN**. The screen would then display lines 13 through 35 of the file similar to the example display below.

```
13: The specified line is listed first in the range.
14: The current line remains unchanged by the L command.
15:*The current line contains an asterisk.
 .
 .
 .
35: CTRL-BREAK exits interline insert command mode.
```

Listing a Group of Lines Centered around the Current Line

To display a group of 23 lines centered around the current line of the file you are editing, you may simply enter **L** and press **RETURN**. For example, if you are editing a file and the current line is line number 15, you could display lines 4 through 26 simply by entering L and pressing RETURN. The screen would then display the group of lines similar to the example shown below.

```
 4: Delete and Insert
 5: (The D and I commands)
 .
 .
 .
```

```
13: The current line is listed in the middle of the range.
14: The current line remains unchanged by the L command.
15:*The current line contains an asterisk.
 .
 .
 .
26: CTRL-BREAK exits the EDLIN interline insert mode.
```

QUIT EDIT Command

Use the QUIT EDIT command to terminate EDLIN and return to MS-DOS COMMAND.COM *without* saving the file you created or edited. The QUIT EDIT command is invoked at the EDLIN prompt by entering

`Q`
or
`QUIT`

and pressing **RETURN**. The command may not be entered in any other form and does not accept any optional parameters.

When you enter the QUIT EDIT command, the screen displays

```
Abort edit (Y/N)?
```

If you *do* wish to terminate EDLIN without saving changes to disk, enter Y. The system prompt will be displayed.

If you *do not* wish to abort edit and want either to continue your work session or to END EDIT and save changes, then enter **N** (or any other noncontrol or nonfunction key). The EDLIN prompt (*) will be displayed.

Normally, when you use the END EDIT command, the file you have created or edited during the session is written to disk according to the file specification you entered when you invoked EDLIN. If you were editing an existing file, the original (source) file is renamed to have the extension .BAK to indicate that it is a backup file. (When you use the QUIT EDIT command, no changes are saved to disk and no .BAK file is created. However, if you were *editing* an existing file and invoked the QUIT EDIT command, the original file would remain on disk, unchanged. If you were *creating* a file and invoked the QUIT EDIT command, no file would be created on disk.)

REPLACE TEXT Command

Use the REPLACE TEXT command to replace all occurrences of a specifed string within a given range of lines with a second specified string. This command is entered at the EDLIN prompt in the form

```
[line1]|,line2]|?|R[string1]F6|string2]
```

where ***line1*** is the first line of the range of lines in which you want the string replacement made,
line2 is the last line of the range of lines in which you want the string replacement made,
? is an optional parameter used to cause EDLIN to query you before each string replacement is made,
string1 is the string you want replaced, and
string2 is the string you want inserted in place of *string1*.

Line1 and *line2* must be separated by a comma. *String1* and *string2* must be separated by **F6** or **CTRL-Z**. No spaces should be used between the strings unless a space is a part of the string itself.

When you enter a REPLACE TEXT command and press RETURN, EDLIN begins searching the specified range of lines for a match with *string1*. As each occurrence of the string is found, it is replaced with the string you specified for *string2*. The changed line is then displayed on the screen. Only lines in which strings are replaced will be displayed. Lines that do not contain a match for *string1*, and thus are not changed, will not be displayed.

If you included the optional parameter ? in the command line, EDLIN prompts you to accept or reject each changed line. Each time *string1* is located and replaced with *string2* and the changed line displayed, EDLIN stops and prompts

```
O.K.?
```

If you wish to include the changed line in your file, enter **Y** or press **RETURN**. The changed line (that is, the line with *string2* in place) will be retained in your file and EDLIN will go on to the next match for *string1*.

If you do not wish to have the line included in your file as changed, enter **N** (or any nonfunction or noncontrol key other than Y or RETURN). The line will be retained in your file without the change (that is, with *string1* in place), and EDLIN will go on to the next occurrence of *string1*.

If string1 occurs more than once in any given line, EDLIN treats each occurrence of the string individually, and, if you use the ? option, EDLIN will prompt you `O.K.?` for each occurrence of string1. This enables you to replace the string selectively and to prevent the replacement of embedded strings.

When all occurrences of string1 in the specified range of lines have been found, the EDLIN prompt is displayed again.

You may enter the REPLACE TEXT command without specifying all possible parameters. If you do not enter a value for *line1* (in other words, if you enter ***,line2*[?]R*string1*F6*string2*** and press **RETURN**), the first line defaults to the line after the current line.

If you do not enter a value for *line2* (that is, if you enter ***line1*[?]R*string1*F6*string2*** and press **RETURN**), the second line defaults to **#**, or the line following the last line in the file.

If you do not enter a value for *string2* (that is, if you enter ***line1,line2*R*string1*F6** and press **RETURN**), *string1* will be deleted from the lines in the specified range; no replacement will be made. The REPLACE TEXT operation is terminated immediately if you do not enter *string1*.

Using the REPLACE TEXT Command

Suppose you have the following file loaded and ready for edit:

```
1: This is a sample file.
2: Use: to demonstrate dynamic line numbers
3: See what happens when you
4: fool around with
5: those very useful commands that
6: perform the two major editing functions,
7: Delete and Insert
8: (The D and I commands)
9: (Use CTRL-BREAK to exit the EDLIN interline insert mode)
10: Line numbers
11: The insert command can place new lines
12: anywhere in the file; there are no space problems
13: because the line numbers are dynamic;
14: They'll slide all the way to 65533.
```

Suppose you wanted to replace all occurrences of the string "and" with "or" in lines 2 through 12. To do this, you would enter

```
2,12 RandF6or
```

and press **RETURN**. The resultant display of changed lines would be

```
 5: those very useful commors that
 7: Delete or Insert
 8: (The D or I commands)
 8: (The D or I commors)
11: The insert commor can place new lines
```

Notice that, because the REPLACE TEXT command was entered with no spaces around the specified strings, EDLIN replaced the occurrences of "and" within words as well as replacing the word "and" itself. Also notice that line 8 is displayed twice, once for each occurrence of the string "and".

To replace only selected occurrences of a string, you must use the ? parameter in the REPLACE TEXT command line. Suppose you had the original sample file first described in this section. To replace selected occurrences of the string "and" in lines 2 through 12 with "or", you would enter

```
2,12?RandF6or
```

and press **RETURN**. EDLIN would display the first line to be changed and prompt you as follows:

```
5: those very useful commors that
O.K.?
```

Enter **N.** EDLIN leaves the line in its original form and then displays the next line in which the string "and" was found and changed.

```
7: Delete or Insert
O.K.?
```

Enter **Y**. The changed line is saved and the next line displayed is

```
8: (The D or I commands)
O.K.?
```

Enter **Y.** The screen displays

```
8: (The D or I commors)
O.K.?
```

Enter **N**. The last word in the line is left in its original form ("commands") while the first change made is retained. The screen then displays

```
11: The insert commor can place new lines
O.K.?
```

Enter **N**. The line is left in its original form and the EDLIN prompt (*) is displayed. If you were to list all the lines in the file after making the above selected changes, the file would appear as follows:

```
 1: This is a sample file.
 2: Use: to demonstrate dynamic line numbers
 3: See what happens when you
 4: fool around with
 5: those very useful commands that
 6: perform the two major editing functions,
 7: Delete or Insert
 8: (The D or I commands)
 9: (Use CTRL-BREAK to exit the EDLIN interline insert mode)
10: Line numbers
11: The insert command can place new lines
12: anywhere in the file; there are no space problems
13: because the line numbers are dynamic;
14: They'll slide all the way to 65533.
```

SEARCH LINES Command

The SEARCH LINES command enables you to find a given point within a file if you do not know the line number (which would enable you to use the EDIT LINE command) but do know a string for which you can direct EDLIN to search within a specified range of lines. The SEARCH LINES command is entered at the EDLIN prompt in the form

[*line1*] [,*line2*] [?]S*string*

where ***line1*** is the first line in the range of lines you want searched,

line2 is the last line of the range of lines you want searched,

? is an optional parameter causing EDLIN to query you when a match for the specified string is found, and

string is the text or data string you want EDLIN to find.

If the ? option is not used, the search function terminates as soon as a match for the *string* you specify is found. The first line containing a match for the string is displayed and becomes the current line. If no match for *string* is found in the range of lines you specify, the search function terminates and the screen displays

```
Not found
```

If you include the ? option in a SEARCH LINES command line, EDLIN locates the first line in which a match for *string* is found, displays the line, and then prompts

```
O.K.?
```

If you enter **Y** or **RETURN** at this prompt, the search function terminates. The displayed line becomes the current line.

If you enter **N** (or any other nonfunction or noncontrol key other than Y or RETURN), EDLIN continues searching for the string within the range of lines you specified. If/when another match is found, the line containing the match is displayed and EDLIN again prompts `O.K.?`. This process continues either until you enter **Y** at the `O.K.?` prompt and the displayed line becomes the current line, or until the search function terminates because no other match is found. If no match is found within the specified range of lines, the screen displays

```
Not found
```

You may enter the SEARCH LINES command without specifying all possible parameters. If you do not enter a value for line1 (in other words, if you enter ***,line2Sstring*** and press **RETURN**), the first line of the range to be searched defaults to the line after the current line.

If the second line number is omitted from the command line (that is, if you enter ***line1,Sstring*** or ***line1Sstring*** and press **RETURN**), the second line defaults to #, or the line following the last line in the file. Thus, if a command were entered in this form, EDLIN would start at the specified line and search for the specified string through the end of the file.

You must always enter the *string* parameter in the SEARCH LINES command. If you fail to specify the string, the search function is terminated immediately.

Searching for a Single String Occurrence

Assume that the following file exists and is loaded for editing:

```
 1: This is a sample file.
 2: Use: to demonstrate dynamic line numbers
 3: See what happens when you
 4: fool around with
 5: those very useful commands that
 6: perform the two major editing functions,
 7: Delete and Insert 8: (The D and I commands)
 9: (Use CTRL-BREAK to exit the EDLIN interline insert mode)
10: Line numbers
11: The insert command can place new lines
12: anywhere in the file; there are no space problems
13: because the line numbers are dynamic;
14: They'll slide all the way to 65533.
```

To search lines 2 through 12, inclusive, of the file for the first occurrence of the string "and", you would enter

```
2,12Sand
```

and press **RETURN**. The search function will begin, and then stop at the first match found. The screen displays the line containing the match and the EDLIN prompt as follows:

```
      5: those very useful commands that
*_
```

Notice that, since you did not leave any spaces around the string you specified (identifying it as a word), EDLIN found the string within a word, "comm*and*s".

If you wish to repeat the SEARCH LINES command you can either reenter a command line at the EDLIN prompt or use the MS-DOS editing keys to retrieve the command you previously entered from the template and edit it if required. The SEARCH LINES command should be edited so that the range of lines to search begins with the line after the first occurrence of *string* found. (For the above example, you might want the second SEARCH LINES command to be **6,12Sand**. Otherwise, if you entered the same command again, line 5 would be displayed again.) The edited command line is then entered by pressing **RETURN**. (See Intraline Editing Functions in this chapter.) EDLIN then displays the next occurrence found as follows:

```
     7: Delete and Insert
*_
```

Searching for Multiple String Occurrences

An easier way to search for multiple occurrences of a specified string is to use the ? option when you enter the SEARCH LINES command. Suppose you are using the same sample file as shown in the preceding example and again wish to search lines 2 through 12, inclusive, for the string "and". This time, to make it easier to repeat the search function until the correct occurrence of the string is found, you could enter

```
2,12?Sand
```

and press **RETURN**. The search function will begin, and then stop at the first match found and display the line, prompting you as follows:

```
5: those very useful commands that
O.K.?
```

Assume that you do not wish to make this line the current line, and wish to continue the search. Enter **N**. The search function continues, and then the next line containing a match is displayed:

```
7: Delete and Insert
O.K.?
```

Enter **Y**. The search function is terminated and the EDLIN prompt is displayed.

WRITE LINES Command

The WRITE LINES command is used to write lines from the edit buffer to the disk file you are editing, enabling you to clear buffer and working memory space so that you can continue editing another section of the file. The command is entered at the EDLIN prompt in the form

```
[n]W
```

where *n* is the number of lines you wish to have removed from the buffer and written to disk according to the file specification you entered when you invoked EDLIN.

When you are editing a large file that will not fit into working memory (the edit buffer) all at one time and when you have finished editing the current contents of the edit buffer, you *must* use the WRITE LINES command before you can add new lines to the buffer with the APPEND LINES command.

The *n* parameter enables you to specify the number of lines you want EDLIN to write to disk. If you do not specify a value for *n* (that is, if you simply enter W at the EDLIN prompt and press **RETURN**), then lines are written out of the buffer, in sequence, to the output file until the buffer is 1/4 full. Lines remaining in the buffer are renumbered starting with 1.

Intraline Editing Functions

Overview

To edit the contents of a given line, you can use the intraline editing functions provided by the MS-DOS editing and control keys. These are summarized in Table 12.2, and individually described in detail in the remainder of this section.

NOTE: The intraline editing functions are provided by a subset of all the MS-DOS editing and control keys. Refer to Chapter 5, "Command Features," for information on all editing and control keys.

The intraline editing functions make use of a special storage area called the *template*. In EDLIN, the intraline editing functions affect the current line. Thus, when you wish to edit a particular line by using intraline editing functions, you must first make certain that the desired line is the current line. Typically, you would do this by using the EDIT LINE command. (Refer to Interline Commands in this chapter.)

Any line you enter from the keyboard by pressing RETURN is saved in an input buffer called the *template* as well as sent to the system for processing. Thus, the last line you entered is always in the template. When you invoke the EDLIN EDIT LINE command, the contents of the line you specify are automatically copied into the template. With the intraline editing functions, that line may be repeated, changed, or replaced for use in the file you are creating or editing with EDLIN.

Most of the intraline editing functions are invoked by pressing a single key. A few are invoked by pressing two keys in sequence. In the descriptions that follow, the functions are generally referred to by the function name, although the keys used to invoke them are clearly defined, both in the text and in Table 12.2.

Intraline Editing Functions

Table 12.2. Intraline Editing Functions

KEY DESIGNATION	FUNCTION NAME	DESCRIPTION
F1 or -->	COPY1	Copies one character from the template.
F2 *x*	COPYUP	Copies multiple characters from the template, up to the specified character, *x*.
F3	COPYALL	Copies all characters in the template.
DEL	SKIP1	Skips (deletes) one character in the template.
F4 *x*	SKIPUP	Skips (deletes) multiple characters in the template, up to the specified character, *x*.
ESC	QUIT INPUT or VOID	Voids the current input without affecting the template.
INS	INSERT	Invokes the insert mode, such that new input does not overwrite any characters in the template.
F5	NEW TEMPLATE	Creates new template.

NOTE: The INS key is a toggle. The first time the key is pressed, the insert mode is turned on; the second time the key is pressed, the insert mode is turned off. In addition, use of some of the other editing keys will turn the insert mode off automatically.

COPY1 Function

The COPY1 function is invoked from the current line by pressing the **F1** or the --> key. This function copies one character from the template to the line you are editing, and advances the cursor one character position. The character copied is the first character contained in the template when **F1** is pressed; this character is also called the *current character*. Once a character has been copied from the template, the next character in the template becomes available. Thus, you could use the COPY1 command to copy a sequence of characters from the template without changing them in any way.

NOTE: When the **F1** key is pressed, the insert mode is turned off automatically.

Example

Suppose you are editing a file and have invoked the EDIT LINE command for line number 1. Assume that the screen then displays line 1 as follows:

```
1: *This is a sample file.
1: *_
```

When you made line 1 the current line with the EDIT LINE command, the contents of the line were copied to the template as well as displayed on the screen. Suppose you wished to copy one character from the template to the line being edited. You would do this by pressing **F1**. The screen would then display

```
1: *This is a sample file.
1: *T_
```

Each time **F1** is pressed, another character is copied from the template. Thus, if you pressed F1 again, the screen would display

```
1: *This is a sample file.
1: *Th_
```

If you pressed F1 two more times, the screen would then display

```
1: *This is a sample file.
1: *This_
```

COPYUP Function

This command copies multiple characters from the template, starting with the first character in the template and stopping when the specified character is encountered. The COPYUP function is invoked from the current line by pressing the **F2** key and then pressing the character key (designated by *x* in Table 12.2) corresponding to the character up to which you wish to copy. The specified character is not copied. If the template does not contain the specified character, or if no character is specified, nothing is copied.

NOTE: When the F2 key is pressed, the insert mode is turned off automatically. (Refer to INSERT Function.)

Example

Suppose you are editing a file and have invoked the EDIT LINE command for line number 1. Assume that the screen then displays

```
1: *This is a sample file.
1: *_
```

When you made line 1 the current line, the contents of the line were moved to the template as well as displayed on the screen for edit. To copy the contents of the template up to, but not including, the letter p, you would press **F2** and then press **P**. The screen would then display

```
1: *This is a sample file.
1: *This is a sam_
```

COPYALL Function

The COPYALL function copies all characters in the template to the line being edited. The function is invoked by pressing **F3**. When all characters have been copied to the current line, the cursor is positioned just after the last character of the line, regardless of what the cursor position was when F3 was pressed.

NOTE: When the F3 key is pressed, the insert mode is turned off automatically.

Example

Suppose you are editing a file and have invoked the EDIT LINE command to make line number 1 the current line. Assume that the screen then displays line 1 as follows:

```
1: *This is a sample file.
1: *_
```

When you made line 1 the current line with the EDIT LINE command, the contents of the line were copied to the template as well as displayed on the screen. If you press **F3** at this point, the entire template would be copied to the current line, causing the screen to display

```
1: *This is a sample file.
1: *This is a sample file._
```

SKIP1 Function

Just as you can copy one character at a time from the template, you can skip or delete one character at a time from the template. The SKIP1 function skips (deletes) the current character in the template. The character is not copied to the input buffer or to the current line.

The SKIP1 function is invoked by pressing **DEL**. This command does not terminate the insert mode, if the insert mode is active when the key is pressed.

Example

Suppose you are editing a file and have invoked the EDIT LINE command to make line number 1 the current line. Assume that the screen then displays the line as follows:

```
1: * This is a sample file.
1: *_
```

When you made line 1 the current line with the EDIT LINE command, the contents of the line were copied to the template as well as displayed on the screen. To delete the first character from the template, you would press **DEL**. The screen display does not change, but the first character is deleted from the template. To demonstrate that the character was in fact deleted, you could press **F3** (invoking the COPYALL function). The screen would then display

```
1: *This is a sample file.
1: *his is a sample file._
```

SKIPUP Function

Just as you can copy multiple characters from the template by using the F2 key and specifying the character you wish to copy up to, you can skip (delete) multiple characters from the template by using the SKIPUP function.

The SKIPUP function is invoked by pressing the **F4** key and then pressing the character key (designated by *x* in Table 12.2) corresponding to the character to which you wish to skip. If the template does not contain the specified character, no characters are skipped (deleted). This function does not terminate the insert mode, if the insert mode is on when the F4 key is pressed.

Example

Suppose you are editing a file and have invoked the EDIT LINE command for line number 1. Assume that the screen then displays line 1 as follows:

```
1: *This is a sample file.
1: *_
```

When you made line 1 the current line, the contents of the line were moved to the template as well as displayed on the screen for editing. To skip the characters in the template up to, but not including, the letter p in "sample", you would press **F4** and then press the **P** key. The screen display does not change, but the characters occurring before the letter p in the template are deleted. To demonstrate this, you could invoke the COPYALL function by pressing **F3**. The screen would then display

```
1: *This is a sample file.
1: *ple file._
```

Notice that the cursor is moved to the end of the edited line.

QUIT INPUT or VOID Function

The QUIT INPUT function enables you to cancel a line you are typing without affecting the contents of the template. This commands flushes the contents of the input buffer (empties the buffer). The template is not affected, because characters you type are not sent to the template until you press RETURN.

The QUIT INPUT function is invoked by pressing the **ESC** key. A backslash (\) is displayed at the end of the line to show that it was cancelled, and a return and line feed are output to move the cursor to a new, blank line. You may then press **F3 RETURN** to leave the line unchanged, type in a new line, or use the intraline editing functions to edit the existing line.

NOTE: If the insert mode is on when the QUIT INPUT function is invoked, the insert mode is turned off.

Example

Suppose you are editing a file and have invoked the EDIT LINE command to make line number 1 the current line. Assume the screen then displays line 1 as follows:

```
1: *This is a sample file.
1: *_
```

Suppose you start to enter new text for the line by typing in **Sample file** but do not press **RETURN**. The screen would then appear as follows:

```
1: *This is a sample file.
1: *Sample file_
```

As long as you have not pressed **RETURN**, you can cancel the line you are entering by pressing **ESC**. Assume you did this in the example file. The screen would then display

```
1: *This is a sample file.
1: *Sample File\
    _
```

You could resume editing line 1. To demonstrate that the template was not affected by the QUIT INPUT function, you could press **F3** to invoke the COPYALL function. The screen would then display

```
1: *This is a sample file.
1: * Sample File\
1: *This is a sample file._
```

INSERT Function

The INSERT function enables you to insert characters within a line without overwriting characters in the template. This function is invoked by pressing the **INS** key. When the intraline insert mode is turned on, characters are entered in the current line before the cursor position.

Once the INSERT function is invoked, the intraline insert mode stays on until it is turned off. The insert mode is turned off automatically by some of the other intraline editing functions, or may be turned off by pressing the INS key a second time. (The INS key functions as a toggle to turn the intraline insert mode on and off.)

NOTE: The intraline insert mode described here is not to be confused with the EDLIN interline insert mode described under Interline Commands. The intraline insert mode invoked with the INS key enables you to insert characters in the current line without affecting characters in the template. The interline insert mode enables you to add one or more lines to the file.

Example

Suppose you are editing a file and have invoked the EDIT LINE command for line number 1. Assume that the screen displays line 1 as follows:

```
1: *This is a sample file.
1: *_
```

When you made line 1 the current line, the contents of the line were copied to the template as well as displayed on the screen for edit. Using several of the intraline editing functions, you can copy part of the template, invoke the INSERT function to insert characters in the line, and then copy the remainder of the template. This could be done as follows:

- To copy characters in the template up to, but not including, the letter p, press **F2** and then press **P**. The screen displays

  ```
  1: *This is a sample file.
  1: *This is a sam_
  ```

Intraline Editing Functions

- To invoke the INSERT function, press **INS**. Then, type in the letters **son**. The screen displays

  ```
  1: *This is a sample file.
  1: *This is a samson_
  ```

  Notice that the letters that were inserted were inserted in front of the cursor, leaving the cursor at the end of the current line.

- To skip the next three letters in the template (that is, the letters p, l, and e) press the **DEL** key three times. The screen display does not change.

- To copy the remainder of the template to the current line, press the **F3** key. The screen then displays

  ```
  1: *This is a sample file.
  1: *This is a samson file._
  ```

- Press **RETURN** to save line 1 as it appears and go on to the next line.

  NOTE: If you had pressed RETURN after inserting the characters **son**, the remaining characters in the template would have been truncated or deleted, and the template would simply contain the line "This is a samson".

NEW TEMPLATE Function

Use the NEW TEMPLATE function to copy the contents of the input buffer to the template, destroying the existing template contents and creating a new template. This function is invoked by pressing **F5**.

When you press **F5**, the input buffer is flushed (emptied) when its contents are moved to the template. An "at" sign (@) is displayed at the end of the line to show that it has been copied to the template, and a return and line feed are output to move the cursor to a new, blank line. Additional editing can then be done by using the new template. If the intraline insert mode was on, the NEW TEMPLATE function terminates it.

When you create a new template, the current line is *not* changed. The current line number is the same as it was when you pressed **F5**. This is different than if you moved the contents of the current line to the template by pressing **RETURN**.

When you press RETURN at the end of the line, that line is no longer the current line. Instead, the next line becomes the current line or the EDLIN prompt is displayed, depending upon the EDLIN command or function you are using. If you are using the EDLIN INSERT LINE command, for example, the next line number is displayed when you press RETURN. If you are editing a given line by using the EDIT LINE command and press RETURN, the EDLIN prompt will be displayed.

Example

Suppose you are editing a file and have invoked the EDIT LINE command to make line number 1 the current line. Assume the screen then displays line 1 as follows:

```
1: *This is a sample file.
1: *_
```

When you made line 1 the current line, the contents of the line were moved to the template as well as displayed on the screen for edit. By using several of the intraline editing functions, you can copy part of the template, replace some characters, insert some characters, and copy the remainder of the template to create a new line. Then you can use the NEW TEMPLATE function to move the line to the template for additional editing. This could be done as follows:

Intraline Editing Functions

- To copy characters in the template up to, but not including, the letter p, press **F2** and then press **P**. The screen displays

```
1: *This is a sample file.
1: *This is a sam_
```

- Type in the letters **son**. Because the insert mode is not on, this replaces the letters "ple" in the template and the screen displays

```
1: *This is a sample file.
1: *This is a samson_
```

- To invoke the intraline insert mode, press **INS**. Type in the letters **ite**. The screen now displays

```
1: *This is a sample file.
1: *This is a samsonite_
```

- To copy the characters remaining in the template to the current line, press **F3**. The screen displays

```
1: *This is a sample file.
1: *This is a samsonite file._
```

- To create a new template containing the current contents of line 1 without changing the current line, press **F5**. The screen displays

```
1: *This is a sample file.
1: *This is a samsonite file.@
   _
```

You may now continue editing the line using the new template.

Error Messages

There are two categories of error messages that you may receive while using EDLIN. The first category consists of those messages that may be displayed if an error occurs when you are invoking the EDLIN program itself from the system prompt. The second category consists of messages that may be displayed once EDLIN has been successfully invoked and you are editing or creating a file.

Errors when Invoking EDLIN

```
Cannot edit .BAK file--rename file
```

EXPLANATION: This message is displayed when you attempt to invoke EDLIN by using a file specification that includes a .BAK extension. EDLIN will not accept .BAK files for editing because files with this extension are assumed to be backup files.

If you need the .BAK file for editing purposes, you must either rename the file by using the RENAME command or copy the file (using the COPY command) to another file with a different extension. Then, you may invoke EDLIN again by using the renamed file or the file copy.

```
File Creation Error
```

EXPLANATION: This message will be displayed if you included any wildcard characters in the EDLIN command line *filespec* parameter. You may *not* use wildcards when specifying the file to create or edit; only one file may be specified. (You may however, include a valid drive name or directory path name.)

At the system prompt, reenter the EDLIN command line, including a unique file specification for the file to create or edit.

Error Messages

```
File name must be specified
```

EXPLANATION: You entered EDLIN and pressed RETURN without specifying the file to create or edit. You cannot invoke EDLIN without the *filespec* parameter.

When the system prompt is displayed, reenter an EDLIN command line that includes the file specification for the file to be created or edited.

```
Invalid drive or file name.
```

EXPLANATION: This message will be displayed if you entered an invalid drive name, an invalid directory path name, and/or an invalid file name as part of the EDLIN command line *filespec* parameter. The problem is probably due to a syntax or typographical error.

Reenter the EDLIN command line, including a valid file specification in the correct syntax.

```
No end-of-file mark found in file
```

EXPLANATION: This message is displayed when EDLIN did not find an end-of-file mark (1A hex) when loading the file you specified. After loading the file, EDLIN scans from the last line of the file for the end-of-file mark. When it finds the mark, anything following the mark is truncated.

If EDLIN finds no end-of-file mark, it assumes that there is nothing to edit.

When this message is displayed, you should delete the file (using the DEL or ERASE command), then recreate it by entering the command **EDLIN *filespec*.**

```
No room in directory for file
```

EXPLANATION: When you specified a file to be created as part of the EDLIN command line, either the system found that the directory is full, or you specified an invalid drive name or an invalid file name as part of the file specification.

First, check the EDLIN command line you entered for the presence of an invalid drive name or an invalid file name. (If the command line is no longer visible on the screen and it was the last line entered, you may retrieve it from the template by pressing **F3**.) If this type of error is discovered, reenter the EDLIN command line, specifying a valid drive name and/or file name as part of the file specification.

If no error was made in the file specification, run the CHKDSK program for the disk in the specified drive. If the resultant status report shows that the directory is full, remove the disk and use a new disk. (Refer to Chapter 11, "Command Descriptions," for information on the CHKDSK command.)

Errors while Editing

```
Disk Full--file write not completed
```

EXPLANATION: This message occurs when you have entered the END EDIT command to close the editing session and save your file to disk, but the disk does not have enough free space to contain the whole file. The END EDIT write operation is aborted and the system prompt is displayed. Some of the file may have been written to disk before termination. If you were editing rather than creating a file, the backup or original version of the file should still be intact.

When the Disk Full error occurs, only a portion (at most) of your edited file is saved. Check the directory to see whether any of the file was saved. If some of the file exists, you should probably delete it (using the DEL or ERASE command) and complete another editing session. The portion of the file that was not written out to disk is *not* recoverable. For this reason, you should always make certain before you start EDLIN that the disk specified in the file specification (or the default disk, if no drive name is entered) has sufficient space for the file to be written.

Error Messages

```
Entry Error
```

EXPLANATION: This message means that the last EDLIN command you entered contained a syntax error and therefore was not executed. Check the command you entered; then, reenter the command in the correct syntax

```
Line too long
```

EXPLANATION: When you invoked a REPLACE TEXT command, the string you specified as the replacement string causes the line to expand beyond the maximum allowable 253 characters. EDLIN aborts the replace function.

When this occurs, you can divide each line containing the string you want to replace into two lines. Then, reinvoke the REPLACE TEXT command.

Chapter 13
LIB

Purpose

LIB is an MS-DOS utility that enables you to delete modules from a library file, change library modules into object files, and append object files to a library file.

NOTE: This chapter assumes that the user is familiar with Assembly language programming and with the operation of 8086 and 8088 registers and instructions. The user should also know the definition of microcomputer terms such as: bit, byte, flag and register.

Entry Forms

```
LIB
```

where **LIB** invokes the LIB command and begins the prompting.

```
LIB library operations, [listing]
```

where **LIB** invokes the LIB command;
library is the name of a library file with an optional drive name and/or path name specification;
operations is a command character (+, −, *) followed by the module or object file name with an optional drive name and/or path name specification; and
listing is the name of the optional file, with an optional drive name and/or path name specification, to receive the cross-reference listing of PUBLIC symbols in the library modules.

```
LIB @filespec
```

where **LIB** invokes the LIB command;
@ tells LIB that all answers to the prompts are contained in the file name which follows; and
filespec is the name of your response file with an optional drive name and/or path name specification.

Preliminary Concepts

LIB creates and modifies library files that are used with Microsoft's LINK Utility. LIB can add object files to a library, delete modules from a library, or extract modules from a library and place the extracted modules into separate object files.

LIB provides a means of creating either general or special libraries for a variety of programs or for specific programs only. With LIB you can create a library for a compiler, or you can create a library for one program only—which would permit very fast linking.

You can modify individual modules within a library by extracting the modules, making changes, then adding the modules to the library again. You can also replace an existing module with a different module or with a new version of an existing module.

The command scanner in LIB is the same as the one used in Microsoft's LINK, Pascal, FORTRAN, and other 16-bit Microsoft products. If you have used any of these products, using LIB will be familiar to you. LIB entry forms are straightforward, and LIB prompts you for any specifications it needs that you have not supplied.

Overview of LIB Operation

LIB performs three basic actions: it deletes modules from a library file, it changes object files into modules and appends them to a library file, and it extracts modules to create an object file. These three actions are the basis for five library management functions:

- Delete a module.
- Extract a module and place it in a separate object file.
- Append an object file as a module of a library.
- Replace a module in the library file with a new module.
- Create a library file.

During each library session, LIB first deletes or extracts modules, then appends new ones. In a single operation, LIB reads each module into memory, checks it for consistency, and writes it back to the file (Figure 13.1). (In this figure, and the following four figures, the capital letters represent modules.) If you delete a module, LIB reads in that module but does not write it back to the file (Figure 13.2). When LIB writes back the next module to be retained, it places the module at the end of the last module written. This procedure effectively "closes up" the disk space to keep the library file from growing larger than necessary (Figure 13.3).

When LIB has read through the whole library file, it appends any new modules to the end of the file. LIB can then copy this module to an object file (Figure 13.4). Finally, LIB creates the index, which the LINK command uses to find modules and symbols in the library file, and outputs a cross-reference listing of the PUBLIC symbols in the library, if you request such a listing (Figure 13.5). (Building the library index may take some extra time, up to 20 seconds in some cases.)

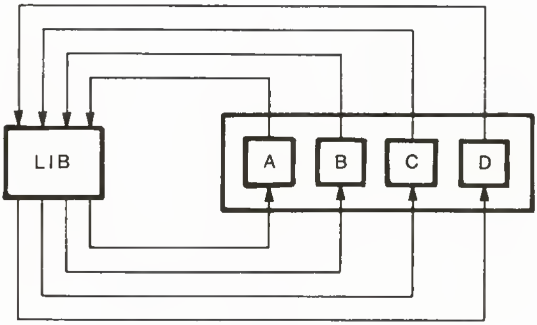


Figure 13.1 Consistency Check Only

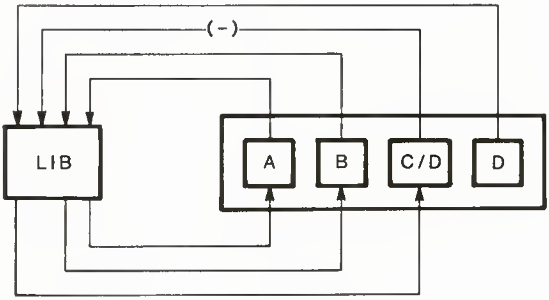


**Figure 13.2 Delete Module C;
Write Module D to Space of C**

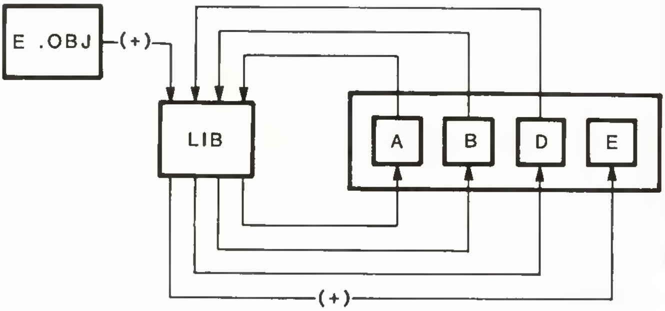


Figure 13.3 Append Object File E.OBJ as Module E

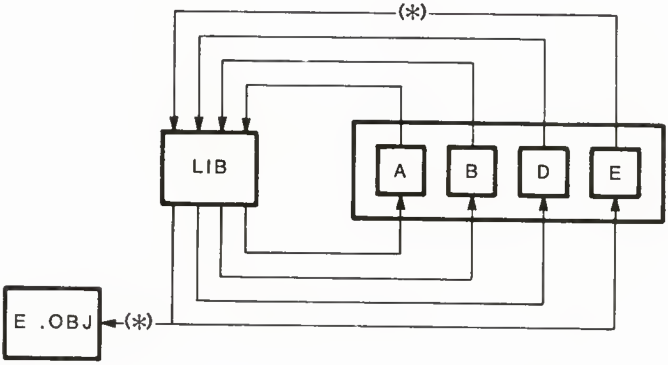


Figure 13.4 Extract Module E; Copy into a Separate Object File; Return to Library

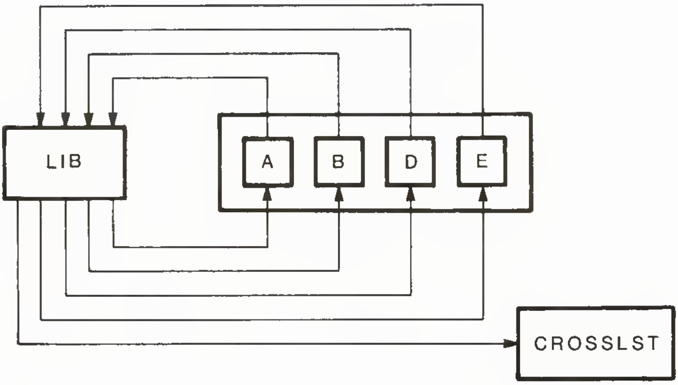


Figure 13.5 Consistency Check, Output Cross-Reference Listing of PUBLIC Symbols

For example, when you type

```
LIB PASCAL-HEAP+HEAP;
```

the library module HEAP is deleted from the library file and the file HEAP.OBJ is added as the last module in the library. This order of execution prevents confusion in LIB when a new version of a module replaces a version in the library file. Note that the replace function is simply the delete and append functions in succession. Also note that you can specify delete, append, or extract functions in any order; the order is insignificant to the LIB command scanner.

Running LIB

Running LIB requires two types of entries: the command to invoke LIB, and answers to LIB command prompts. Usually you will enter all the answers to LIB prompts on the terminal keyboard. Answers can also be contained in a *response file*. A *response file* is a file which contains answers to the LIB command prompts.

LIB recognizes six command characters (+, −, *, ;, &, and CTRL-BREAK). Some are used as a *required* part of LIB commands, and the others *assist* you in entering LIB commands.

Invoking LIB

LIB may be invoked through three methods. By the first method, the Command Prompt Method, you enter answers to individual prompts. By the second method, the Command Line Method, you enter all answers on the line used to invoke LIB. By the third method, the Response File Method, you specify a response file that contains all the necessary answers. This response file must exist before you enter the LIB command.

Summary of Methods to Invoke LIB

Command Prompt Method	`LIB`
Command Line Method	`LIB` ***library operations***`,` [***listing***]
Response File Method	`LIB @`***filespec***

Command Prompt Method

To begin under the Command Prompt Method, type the following command at the system prompt:

```
LIB
```

and press the **RETURN** key. LIB loads into memory. Then, LIB returns a series of three text prompts that appear one at a time. You answer the prompts to perform specific tasks.

Summaries of the LIB command prompts and command characters follow. For a full description, refer to the Command Prompt and Command Character sections later in this chapter.

Running LIB

Summary of LIB Command Prompts

PROMPT	RESPONSES
`LibraryFile:`	List file name of library to be manipulated, with optional drive name and/or path name (Default file name extension: .LIB)
`Operations:`	List command character(s) followed by module name(s) or object file name(s), with optional drive name(s) and/or path name(s), (Default action: no changes—Default object file name extension: .OBJ)
`Listfile:`	List file name for a cross-reference listing file, with optional drive name and/or path name (Default: NUL; no file)

Summary of Command Characters

Character	Action
+	Append an object file as the last module.
–	Delete a module from the library.
*	Extract a module and place in an object file.
;	Use default responses to remaining prompts.
&	Extend current physical line; repeat command prompt.
CTRL-BREAK	Abort library session.

Command Line Method

The Command Line Method enables you to enter all of the specifications for a LIB operation in a single command line.

To begin under the Command Line Method, type the following command line at the system prompt:

```
LIB library operations,[listing]
```

The entries following LIB are responses to the command prompts. The library and operations fields and all operations entries must be separated by one of the command characters—plus, minus, and asterisk (+, −, *). If a cross-reference listing is wanted, the name of the file must be separated from the last operations entry by a comma.

library represents the name of a library file. LIB assumes that the file name extension is .LIB, which you may override by specifying a different extension. You may also specify a drive name and/or path name. If the file name given for the library field does not exist, LIB will prompt you:

```
Library does not exist.  Create?
```

Enter either **Yes** (to create a new library file) or **No** (to abort the library session) and press **RETURN**. Actually, LIB checks the response for the letter Y as the first character. If N is the first letter, LIB terminates and returns to the operating system.

operations include deleting a module, appending an object file as a module, or extracting a module as an object file from the library file. Use the three command characters plus (+), minus (−), and asterisk (*) to tell LIB what action to take with each module or object file. You may specify a drive name and/or path name with each module or object file.

listing represents the name of the file you want to receive the cross-reference listing of PUBLIC symbols in the library modules. The list is compiled after all module manipulation has taken place.

Running LIB

To select the default for remaining field(s), you may enter the semicolon (;) command character.

If you enter a library file name followed immediately by a semicolon, LIB will read through the library file and perform a consistency check. No changes will be made to the modules in the library file.

If you enter a library file name followed immediately by a comma and a list file name, LIB will perform its consistency check of the library file, then produce the cross-reference listing file.

If you have many operations to perform during a library session, use the ampersand (&) command character to extend the line so that you can enter additional object file names and module names. Be sure to always include one of the command characters for operations (+, −, *) before the name of each module or object file name.

Examples

To delete the module HEAP from the library file PASCAL.LIB and append the object file HEAP.OBJ, you would enter

```
LIB PASCAL-HEAP+HEAP;
```

and press **RETURN**. LIB deletes the module HEAP from the library file PASCAL.LIB and appends the object file HEAP.OBJ as the last module of PASCAL.LIB (the module will be named HEAP). The semicolon indicates LIB is to use the default responses for the remaining prompts.

To perform a consistency check of the library file PASCAL.LIB, you would enter

```
LIB PASCAL;
```

and press **RETURN**. LIB performs a consistency check. No other action is performed.

To perform a consistency check and output a cross-reference listing, you could enter

```
LIB PASCAL,PASCROSS.PUB
```

LIB performs a consistency check of PASCAL.LIB and outputs a cross-reference listing file named PASCROSS.PUB.

Response File Method

The Response File Method enables you to enter all of the specifications for a LIB operation in a single file.

To begin under the Response File Method, type the following command line at the system prompt:

LIB *@filespec*

@ is a pointer that tells LIB that all the prompt answers are contained in the file whose name follows.

Filespec represents the name of a response file with optional drive name and path name. A response file contains answers to the LIB prompts (summarized under Command Prompt Method and described fully in the Command Prompts section). The Response File Method permits you to conduct the LIB session without interactive (direct) user responses to the LIB prompts.

NOTE: Before using the Response File Method to start LIB, you must first create the response file. You can create a response file using EDLIN, the MS-DOS editor, the COPY command or a text editor.

A response file consists of text lines, one for each prompt. The responses must appear in the same order as the command prompts appear.

Command characters are used in a response file in the same manner as they are used for answering prompts on the terminal keyboard.

Running LIB

When the library session begins, each prompt will, in turn, be displayed with the responses from the response file. If the response file does not contain answers for all the prompts, LIB will use the default responses. (Default responses—no changes to the modules currently in the library file for the `Operations:` prompt, and no cross reference listing file created for the `List file:` prompt.)

If you enter a library file name followed immediately by a semicolon, LIB will read through the library file and perform a consistency check. No changes will be made to the modules in the library file.

If you enter a library file name and press RETURN, then a comma and press RETURN, and finally, a list file name and press RETURN,

```
PASCAL
,
CROSSLST
```

LIB performs its consistency check of the library file, and produces the cross-reference listing file.

To create a response file which deletes a module, extracts a module and places it in an object file, and finally appends that object file, you could enter

```
PASCAL
+CURSOR+HEAP-HEAP*FOIBLES
CROSSLST
```

and press **RETURN** at the end of each line. This response file causes LIB to delete the module HEAP from the PASCAL.LIB library file. LIB extracts the module FOIBLES and places it in an object file named FOIBLES.OBJ. LIB then appends the object files CURSOR.OBJ and HEAP.OBJ as the last two modules in the library. Then, LIB will create a cross reference file named CROSSLST.

Command Prompts

LIB is activated by entering responses to three text prompts. When you have entered your response to the current prompt, the next appears. When the last prompt has been answered, LIB performs its library management functions. When the library session is finished, LIB exits to the operating system. When the operating system prompt is displayed, LIB has finished the library session successfully. If the library session is unsuccessful, LIB returns the appropriate error message. (Error messages are explained at the end of this section.)

LIB prompts for the name of the library file, the operation(s) you want to perform, and the name you want to give to a cross reference listing file, if any.

Library File:

Enter the name of the library file (with optional drive name and path name) that you want to manipulate. LIB assumes that the file name extension is .LIB. You can override this assumption by giving a file name extension when you enter the library file name. Because LIB can manage only one library file at a time, only one file name is allowed in response to this prompt. Additional responses, except the semicolon command character, are ignored.

If you enter a library file name and follow it immediately with a semicolon command character, LIB will perform a consistency check only, then return to the operating system. Any errors in the file will be reported.

If the file name you enter does not exist, LIB returns the prompt:

```
Library does not exist.  Create?
```

You must enter either **Yes** or **No**, and press **RETURN**. Actually, LIB checks the response for the letter Y as the first character. If N is the first letter, LIB terminates and returns to the operating system.

Running LIB

Operations:

Enter one of the three command characters for manipulating modules (+, −, *), followed immediately (no space) by the module name or the object file name (with optional drive name and path name). The plus sign appends an object file as the last module in the library file (for further discussion, refer to the plus sign description under the Command Character section). The minus sign deletes a module from the library file. The asterisk extracts a module from the library and places it in a separate object file with the file name taken from the module name and file name extension .OBJ.

When you have a large number of modules to manipulate (more than can be typed on one line), enter an ampersand (&) as the last character on the line. LIB will repeat the `Operations:` prompt, which permits you to enter additional module names and object file names.

LIB allows you to enter operations on modules and object files in any order you want.

More information about order of execution and what LIB does with each module is given in the descriptions of each command character.

List file:

If you want a cross-reference list of the PUBLIC symbols in the modules in the library file after your manipulations, enter a file name in which you want LIB to place the cross-reference listing (with optional drive name and path name). If you do not enter a file name, no cross-reference listing is generated (a NUL file).

The response to the `List file:` prompt is a file specification. Therefore, you can specify, along with the file name, a drive (or device) name and a file name extension. The list file is not given a default file name extension. If you want the file to have a file name extension, you must specify it when entering the file name.

The cross-reference listing file contains two lists. The first list is an alphabetical listing of all PUBLIC symbols. Each symbol name is followed by the name of its module. The second list is an alphabetical list of the modules in the library. Under each module name is an alphabetical listing of the PUBLIC symbols in that module.

Command Characters

LIB provides six command characters. Three of the command characters are required in responses to the `Operations:` prompt; the other three command characters provide you with helpful commands to LIB.

+ The plus sign followed by an object file name appends the object file as the last module in the library named in reponse to the `Library File:` prompt. When LIB sees the plus sign, it assumes that the file name extension is .OBJ. You may override this assumption by specifying a different file name extension.

LIB uses only the file name portion of the file specification for the module name. For example, if the object file to be appended as a module to a library is

B:CURSOR.OBJ

and you respond to the `Operations:` prompt with

```
+B:CURSOR.OBJ
```

then LIB will create a module named CURSOR in the library.

– The minus sign followed by a module name deletes that module from the library file. LIB then "closes up" the file space left empty by the deletion. This cleanup action keeps the library file from growing larger than necessary with empty space. Remember that new modules, even replacement modules are added to the end of the file, not stuffed into space vacated by the deleted modules.

* The asterisk followed by a module name extracts that module from the library file and places it into a separate object file. (Even though the module is copied to a separate object file, the module will still exist in the library.) The module name is used as the file name. LIB adds the default drive name and the file name extension .OBJ. For example, if the module to be extracted is

CURSOR

and the current default disk drive is A, and you respond to the `Operations:` prompt with

```
*CURSOR
```

LIB will then extract the module named CURSOR from the library file and set it up as an object file with the file specification of

A:CURSOR.OBJ

(The drive name and file name extension cannot be overridden. You can, however, rename the file, giving a new file name extension, and/or copy the file to a new disk drive, giving a new file name and/or file name extension.)

Use a single semicolon (;) followed immediately by a RETURN at any time after responding to the first prompt (from `Library File:` on) to select default responses to the remaining prompts. This feature saves time and eliminates the need to answer additional prompts.

NOTE: Once the semicolon has been entered, you can no longer respond to any of the prompts for that library session. Therefore, do not use the semicolon to skip over some prompts. For this, press the RETURN key.

If you enter

```
Library file:  FUN
Operations:   +CURSOR;
```

the remaining prompt will not appear, and LIB will use the default value (no cross-reference file).

& Use the ampersand to extend the current physical line. This command character will only be needed for the `Operations:` prompt. LIB can perform many functions during a single library session. The number of modules you can append, replace, or extract is limited only by disk space. The number of modules you can delete is limited only by the number of modules in the library file.

The line length for a response to any prompt is limited to the line length of your system. For a large number of responses to the `Operations:` prompt, place an ampersand at the end of a line and press **RETURN**. LIB will display the `Operations:` prompt again, then enter more responses.

For example, when the ampersand is entered after FOIBLES (the end of the second line) and you press **RETURN**, Lib displays the `Operations:` prompt

```
Library file:  FUN
Operations:   +CURSOR-HEAP+HEAP*FOIBLES&
Operation:
```

If you enter

```
*INIT+ASSUME+RIDE;
```

and press **RETURN**, LIB will delete the module HEAP, extract modules FOIBLES and INIT (creating two files, FOIBLES.OBJ and INIT.OBJ); then it will append the object files CURSOR, HEAP, ASSUME, and RIDE to the library file FUN.

Note, however, that LIB allows you to enter your `Operations:` reponses in any order. You may use the ampersand character as many times as you need.

CTRL-BREAK Use CTRL-BREAK at any time to abort the library session. If you enter an erroneous response, such as the wrong file name or module name, or an incorrectly spelled file name or module name, you must enter CTRL-BREAK to exit LIB, then reinvoke LIB and start over. (If the error has been typed but not entered, you may delete the erroneous characters, but for that line only.)

Error Messages

Most errors cause the LIB session to abort. Therefore, after the cause is determined and corrected, LIB must be rerun.

`symbol is a multiply defined PUBLIC.  Proceed?`

EXPLANATION: Two modules define the same PUBLIC symbol. You are asked to confirm the removal of the definition of the old symbol. A **No** response leaves the library in an undetermined state. Remove or rename the PUBLIC declaration from one of the source modules and recompile or reassemble.

`Allocate error on VM.TMP`

EXPLANATION: This error occurs because the disk does not have enough space left. Try another disk that contains more space, or free up some space on the disk, and rerun LIB.

`Cannot create extract file`

EXPLANATION: There is not enough room in the disk directory for the extract file. Use a disk which contains less files or delete any unnecessary files from the full disk.

`Cannot nest response file`

EXPLANATION: The response file you specified as containing all your responses invokes another response file. Put all your responses into one file and reinvoke LIB.

`Cannot open VM.TMP`

EXPLANATION: There is not enough room in the disk directory for the VM.TMP file. Use a disk which contains less files or delete any unnecessary files from the full disk.

`Cannot write library file`

EXPLANATION: This error occurs because the disk does not have enough space left. Try another disk that contains more space, or free up some space on the disk, and rerun LIB.

`Close error on extract file`

EXPLANATION: This error occurs because the disk does not have enough space left. Try another disk that contains more space, or free up some space on the disk, and rerun LIB.

`Fatal Error: An internal error has occurred.`

EXPLANATION: Contact Software Consultation at Microsoft, Inc.

`Fatal Error: Cannot create list file`

EXPLANATION: An invalid name was used for the list file, or there is not enough room in the disk directory for the library file. Change the list file name, or use a disk which contains less files or delete any unnecessary files from the full disk.

`Fatal Error: Cannot open input file`

EXPLANATION: The object file name was mistyped or the module does not exist. Examine the directory to determine the correct file name and rerun LIB with that file name.

`Fatal Error: Cannot open response file`

EXPLANATION: The response file name was mistyped or the module does not exist. Examine the directory to determine the correct file name and rerun LIB with that file name.

LIB

Error Messages

```
Fatal Error: Invalid object module/library
```

EXPLANATION: This error occurs because either the object module or library does not exist under the name entered, or it is a bad object module and/or library. The usual cause is a bad disk. Assemble the modules and rerun LIB.

```
Fatal Error: Module is not in the library
```

EXPLANATION: An attempt was made to delete a module that is not currently in the library. Obtain a listing of the library module and rerun LIB using the correct module name.

```
Fatal Error: Write error on library/extract file
```

EXPLANATION: This error occurs because the disk does not have enough space left. Try another disk that contains more space, or free up some space on the disk, and rerun LIB.

```
Input file read error
```

EXPLANATION: This error occurs because of a bad object module or faulty disk. Reassemble the offending module and rerun LIB.

```
Library Disk is full
```

EXPLANATION: This error occurs because there is no more room on the disk. Try another disk that contains more space, or free up some space on the disk, and rerun LIB.

```
Listing file write error
```

EXPLANATION: This error occurs because the disk does not have enough space left. Try another disk that contains more space, or free up some space on the disk, and rerun LIB.

```
No library file specified
```

EXPLANATION: This error occurs because no response was given to the `Library File:` prompt. Run LIB and enter a correct file name to this prompt.

`Read error on VM.TMP`

EXPLANATION: The disk is not able to be read. Attempt this operation again and if you still receive this message, you may have a bad disk.

It is also possible that you removed the disk containing the VM.TMP from the accessed drive.

`Symbol table capacity exceeded`

EXPLANATION: This error occurs because there are too many PUBLIC symbols (about 30K characters in symbols). Either remove any unwanted modules or start creating another library to be used in conjunction with the existing library.

`Too many object modules`

EXPLANATION: The number of object modules exceeds the 500 limit. Remove the unwanted object modules from the library and rerun LIB.

`Too many PUBLIC symbols`

EXPLANATION: The number of PUBLIC symbols exceeds the 1024 limit. Remove the unwanted PUBLIC symbols from the library and rerun LIB.

`Write error on VM.TMP`

EXPLANATION: This error occurs because the disk does not have enough space left. Try another disk that contains more space, or free up some space on the disk, and rerun LIB.

Chapter 14
LINK

Purpose

LINK is an MS-DOS utility that enables you to combine several object modules into one relocatable load module, or run file.

NOTE: This chapter assumes that the user is familiar with Assembly language programming and with the operation of 8086 and 8088 registers and instructions. The user should also know the definition of microcomputer terms such as: bit, byte, flag and register.

Entry Forms

```
LINK
```

where **LINK** invokes the LINK command and begins prompting.

```
LINK object-list,runfile,listfile,lib-list[/x]
```

where **LINK** invokes the LINK command;
object-list is a list of object modules with optional drive name and/or path name specification;
runfile is the file to receive executable output with optional drive name and/or path name specification;
listfile is the file to receive the listing with optional drive name and/or path name specification;
lib-list is a list of library modules to be searched with optional drive name and/or path name specification; and
/x represents any one of the following optional switches:

/DSALLOCATE	Load data at high end of data segment
/HIGH	Place run file as high as possible in memory
/LINE NUMBER	Include line numbers in list file
/MAP	List all global symbols with definitions
/PAUSE	Pause linker session
/STACK:number	Set fixed stack size in run file.
/NO	No default library search.

Entry Forms

```
LINK @filespec
```

where **LINK** invokes the LINK command;
@ tells LIB that all the prompt answers are contained in the file name which follows; and
filespec is the name of your response file with optional drive name and/or path name specification.

Preliminary Concepts

LINK is a relocatable linker designed to link together separately produced modules of 8086 and 8088 object code. The object modules must be 8086 or 8088 files only.

LINK uses prompts for all the necessary and optional commands.

The output file from LINK (called a run file) is not bound to specific memory addresses and, therefore, can be loaded and executed at any convenient address by the operating system.

LINK uses a dictionary-indexed library search method, which substantially reduces link time for sessions involving library searches.

LINK is capable of linking files totaling 900K bytes.

Overview of LINK Operation

LINK combines several object modules into one relocatable load module, or run file. As it combines modules, LINK resolves external references between object modules and can search multiple library files for definitions for any external references left unresolved.

LINK also produces a list file that shows external references resolved and any error messages.

LINK uses available memory as much as possible. When available memory is exhausted, LINK then creates a disk file and becomes a virtual linker.

Figure 14.1 illustrates the LINK operation.

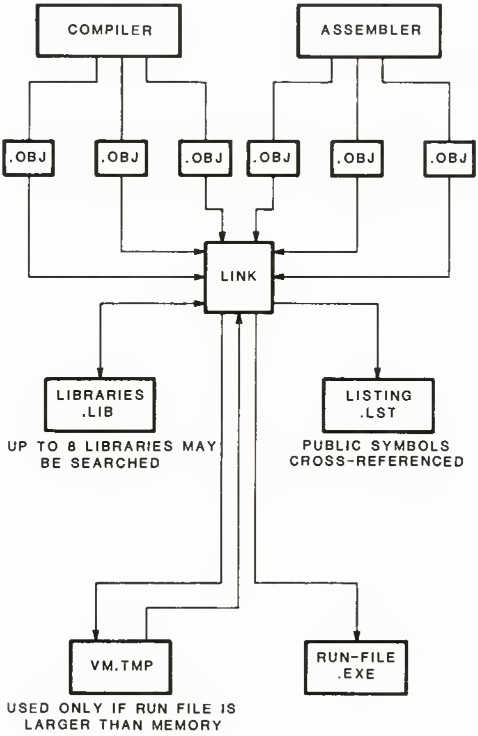

COMPILER
ASSEMBLER
.OBJ
.OBJ
.OBJ
.OBJ
.OBJ
.OBJ
LINK
LIBRARIES .LIB
UP TO 8 LIBRARIES MAY BE SEARCHED
LISTING .LST
PUBLIC SYMBOLS CROSS-REFERENCED
VM.TMP
USED ONLY IF RUN FILE IS LARGER THAN MEMORY
RUN-FILE .EXE

Figure 14.1 LINK Operation

Definitions

The following terms describe the intrinsic operation of LINK. An understanding of the concepts that these terms define provides a basic understanding of the way LINK works.

Of the terms below, the first three appear frequently in LINK error messages.

1. *Segment*

   A *segment* is a contiguous area of memory up to 64K bytes (where K stands for 1024 bytes) in length. A segment may be located anywhere in 8086 memory on a "paragraph" (16 byte) boundary. The contents of a segment are addressed by a segment-register/offset pair.

2. *Group*

   A *group* is a collection of segments which fit within 64K bytes of memory. The segments are included in the group by the assembler, by the compiler, or by you. You give the group name in the Assembly language program. For the high-level languages such as BASIC, FORTRAN, COBOL, and Pascal, the naming is carried out by the compiler.

   The group is used for addressing segments in memory. Each group is addressed by a single segment register. The segments within the group are addressed by the segment register plus an offset. LINK checks to see that the object modules of a group meet the 64K byte constraint.

3. *Class*

   A *class* is a collection of segments. The inclusion of segments in a class controls the order and relative placement of segments in memory. The class name is given by you in the Assembly language program. For the high-level languages (BASIC, FORTRAN, COBOL, Pascal), the naming is carried out by the compiler.

The segments are included in a class at compile time or assembly time. The segments of a class are loaded into memory contiguously. The segments are ordered within a class in the order LINK encounters the segments in the object files. One class precedes another in memory only if a segment for the first class precedes all segments for the second class in the input to LINK.

4. *Alignment*

   Alignment refers to certain segment boundaries. These can be byte, word, or paragraph boundaries.

   Byte Alignment—A segment can begin on any byte boundary.

   Word Alignment—The beginning address of a segment must occur on an even address.

   Paragraph Alignment—The beginning address of a segment must occur on a segment (16-byte) boundary.

5. *Combine Type*

   A *combine type* is an attribute of a segment; it tells the linker how to combine segments of a link name, or it relays other information about the properties of a segment. Combine types are: stack, public, private, and common. The way LINK arranges these combine types is discussed in the next section.

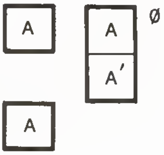
Segment Combination and Arrangement

LINK works with four combine types which are declared in the source module for the assembler or compiler: private, public, stack, and common. (The memory combine type available in

Microsoft's MACRO-86 is treated the same as public. LINK does not automatically place memory combine type as the highest segments.)

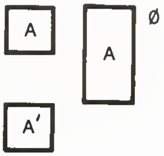
LINK combines segments for these combine types as follows:

Private



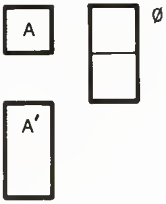
Private segments are loaded separately and remain separate. They may be physically, but not logically, continuous, even if the segments have the same name. Each private segment has its own base address.

Public



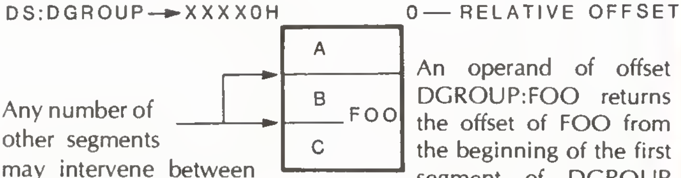
Public segments of the same name and class name are loaded contiguously. Offset is from beginning of first segment loaded through last segment loaded. There is only one base address for all public segments of the same name and class name. (Combine types stack and memory are treated the same as public. However, the stack pointer is set to the last address of the last stack segment.)

Common



Common segments of the same segment name and class name are loaded overlapping one another. There is only one base address for all common segments of the same name. The length of the common area is the length of the longest segment.

Placing segments in a group in the assembler provides offset addressing of items from a single base address for all segments in that group.



Any number of other segments may intervene between segments of a group. Thus, the offset of FOO may be greater than the combined size of segments in the group, but no larger than 64K.

An operand of offset DGROUP:FOO returns the offset of FOO from the beginning of the first segment of DGROUP (segment A here).

Segments are grouped by declared class names. LINK loads all the segments belonging to the first class name encountered, then loads all the segments of the next class name encountered, and continues until all classes have been loaded.

If your program contains:	They will be loaded as:
A SEGMENT 'FOO'	'FOO'
B SEGMENT 'BAZ'	A
C SEGMENT 'BAZ'	E
D SEGMENT 'ZOO'	'BAZ'
E SEGMENT 'FOO'	B
	C
	'ZOO'
	D

If you are writing Assembly language programs, you can exercise control over the ordering of classes in memory by writing a dummy module and listing it immediately after the LINK Object Modules prompt. The dummy module declares segments into classes in the order you want the classes loaded.

NOTE: Do not use this method with BASIC, COBOL, FORTRAN, or Pascal programs. Allow the compiler and the linker to perform their tasks in the normal way.

For example, if you create the dummy module

```
A     SEGMENT  'CODE'
A     ENDS
B     SEGMENT  'CONST'
B     ENDS
C     SEGMENT  'DATA'
C     ENDS
D     SEGMENT  STACK   'STACK'
D     ENDS
E     SEGMENT  'MEMORY'
E     ENDS
```

be careful to declare all classes to be used in your program in this module. If you do not, you lose absolute control over the ordering of classes. (The segments are loaded in their order of occurrence and alphabetically by class name.)

Segment Addresses

The 8088 must be able to address all segments in memory. Any 20-bit number can be addressed. The 8088 represents these numbers as two 16-bit numbers; for example, HEX F:12. The F represents a *canonical frame address* and the 12 is the *offset*. The *canonical frame address* is the base address or segment address that contains the segment. An *offset* is the segment's location, offset from the beginning of the canonical frame.

The linker recognizes a segment by its canonical frame address and its offset within the frame.

To convert the segment address F:12 to a 20-bit number, shift the frame address left 4 bits, and add the offset.

Using the above example HEX F:12

```
            F0      (HEX F shifted left 4 bits)
          + 12      (the offset)
           ___
F:12  =    102      (20-bit address)
```

How LINK Assigns Addresses

To assign addresses to segments, LINK

- Orders each segment by segment and class name.
- Assigns a frame address (starting at 0:0) and offset to each segment (based on alignment and size of each segment, assuming they are contiguous). This information is used for resolving relocatable references.

Relocation Fixups

LINK performs relocation fixups, or resolves address references, on four types of references in object modules.

Short Reference

Short references are all self-relative. Thus, the frame address of the target and source frames are the same. LINK will generate the fixup error message

```
Fixup offset exceeds field width
```

under the following conditions:

1. The target and source frame addresses are different.
2. The target is more than 128 bytes before or after the source frame address.

The resulting value of the short reference must fit into one signed byte.

Near Self-Relative References

When near self-relative references are used, the frame address of the target and source frames are the same. LINK will generate the fixup error message under the following conditions:

1. The target and source frame addresses are different.

2. The target is more than 32K before or after the source frame address.

The resulting value of the near self-relative reference must fit into one signed word (16 bits).

Long References

Long references have a target in another frame. The target must be addressable through the canonical frame specified. LINK will generate the fixup error message under the following conditions:

1. The offset of the target within the specified frame is greater than 64K or less than zero.

2. The beginning of the canonical frame of the target is not addressable by the specified frame.

The resulting value of a long reference must be a frame address and offset.

Files That LINK Uses

LINK works with one or more input files, can produce two output files, may create a virtual memory file, and may be directed to search one to eight library files. For each type of file, you may give a three part file specification. The format for LINK file specifications is:

d:pathname filename

where **d:** is the drive name identifying the disk on which the file is located. Permissible drive names for LINK are A: through O:;
pathname is the directory path the system must use (if other than the current directory of the default or specified disk) to locate the desired file or files; and
filename is the primary file name and the extension, if any, identifying the file.

Input Files

If no extensions are given in the input (Object) file specifications, LINK recognizes by default:

FILE	DEFAULT EXTENSION
Object	.OBJ
Library	.LIB

Output Files

LINK appends to the output (run and list) files the following default extensions:

FILE	DEFAULT EXTENSION
Run	.EXE (may not be overridden)
List	.MAP (may be overridden)

VM.TMP File

LINK uses available memory for the link session. If the files to be linked create an output file that exceeds available memory, LINK creates a temporary file and names it VM.TMP. If LINK needs to create VM.TMP, it displays the message:

```
VM.TMP has been created.
Do not change diskette in drive, d:
```

Once this message is displayed, you must not remove the disk from the default drive until the link session ends. If the disk is removed, the operation of LINK is unpredictable, and LINK might return the error message

```
Unexpected end of file on VM.TMP
```

LINK uses VM.TMP as a virtual memory. The contents of VM.TMP are subsequently written to the file you specified at the `run file:` prompt. VM.TMP is a working file only and is deleted at the end of the linking session.

CAUTION: Do not use VM.TMP as a file name for any file. If *you* created a file named VM.TMP on the default drive and *LINK* needed to create the VM.TMP file, LINK would delete your VM.TMP file and create its VM.TMP file. All the contents of your VM.TMP file would be lost.

Running LINK

Running LINK requires two types of entries: the command to invoke LINK, and answers to LINK command prompts. In addition, six switches control alternate LINK features. Usually, you will enter all the answers to LINK on the terminal keyboard. As an option, answers to the command prompts and any switches may be contained in a Response File. Some command characters are provided to assist you while entering linker commands.

Invoking LINK

LINK may be invoked through three methods. By the first method, the Command Prompt Method, you enter answers to individual prompts. By the second method, the Command Line Method, you enter all answers on the line used to invoke LINK. By the third method, the Response File Method, you specify a response file that contains all the necessary answers. The response file must exist before you enter the LIB command.

Summary of Methods to Invoke LINK

Command Prompt Method	`LINK`
Command Line Method	`LINK` ***filenames***`[/x]`
Response File Method	`LINK` ***@filespec***

Command Prompt Method

To begin under the Command Prompt Method, type the following command at the system prompt:

```
LINK
```

and press the **RETURN** key. LINK loads into memory. Then, LINK returns a series of four text prompts that appear one at a time. You answer the prompts to perform specific tasks.

At the end of each line, you may enter one or more switches, each of which must be preceded by a slash mark. If a switch is not included, LINK defaults to not performing the function described in the Summary of Switches chart.

Summaries of command prompts and switches follow. For a full description, refer to the Command Prompts and Switches sections later in this chapter.

Running LINK

Summary of LINK Command Prompts

PROMPT	RESPONSES
`Object Modules [.OBJ]:`	List .OBJ files to be linked, with optional drive name and/or path name, separated by blank spaces or plus signs (+). If a plus sign is the last character entered, the prompt will reappear. (No default: response is required.)
`Run File [`*`Objfile`*`.EXE]:`	List file name, with optional drive name and/or path name, for executable object code. (Default: first object file name.EXE entered in `Object Modules [OBJ]:` prompt.)
`List File [NUL.MAP]:`	List file name, with optional drive name and/or path name, for listing (Default: no .MAP file created.)
`Libraries [.LIB ]:`	List file names to be searched, with optional drive name and/or path name, separated by blank spaces or plus signs (+). If plus sign is last character entered, the prompt will reappear. (Default: no search.)

Summary of Switches

SWITCH	ACTION
/DSALLOCATE	Load data at high end of Data Segment. Required for Pascal and FORTRAN programs.
/HIGH	Place the run file as high as possible in memory. Do not use with Pascal or FORTRAN programs.
/LINENUMBERS	Include line numbers in list file.
/MAP	List all global symbols with definitions.
/PAUSE	Halt linker session and wait for the RETURN key to be pressed.
/STACK: *number*	Set fixed stack size in run file.
/NO	No default library search.

Command Characters

LINK provides three command characters (+, ;, CTRL-BREAK).

+ Use the plus sign (+) to separate entries and to extend the current physical line following the `Object Modules [.OBJ]:` and `Libraries [.LIB]:` prompts. (A blank space may be used to separate object modules.) To enter a large number of responses (each which may also be very long), enter a plus sign and press the RETURN key at the end of the physical line (to extend the logical line). For example, you would enter

```
Object Modules [.OBJ]:FUN TEXT TABLE CARE+
```

Running LINK

Since the plus sign and pressing RETURN is the last entry which follows the prompt, LINK will prompt you for more modules names. When the `Object Modules [.OBJ]:` (or `Libraries [.LIB]:`) prompt appears again, continue to enter responses. When all the modules to be linked have been listed, be sure the response line ends with a module name and not a plus sign before pressing RETURN.

Thus, your completed entries should look like this:

```
Object Modules [.OBJ]:FUN TEXT TABLE CARE+
Object Modules [.OBJ]:FOO+FLIPFLOP+JUNQUE+
Object Modules [.OBJ]:CORSAIR
```

Use a single semicolon (;) followed immediately by RETURN at any time after the first prompt (from `Run File [.EXE]:` on) to select default responses to the remaining prompts.

NOTE: Once the semicolon has been entered, the user can no longer respond to any of the prompts for that link session. Therefore, do not use the semicolon to skip over some prompts. For this, use RETURN.

For example, when you enter

```
Object Modules [.OBJ]:FUN TEXT TABLE CARE
Run Module [FUN.EXE]:;
```

and press **RETURN** after each line, the remaining prompts will not appear. LINK will use the default values.

CTRL-BREAK Use CTRL-BREAK at any time to abort the link session. If you enter an erroneous response, such as the wrong file name or an incorrectly spelled file name, you must press CTRL-BREAK to exit LINK; then reinvoke LINK and start over. (If the error has been typed but not entered, you may delete the erroneous characters, but for the current line only.)

Command Line Method

The Command Line Method enables you to enter all of the specifications for a LINK operation in a single command line.

To begin under the Command Line Method, type a command line in the following form at the system prompt:

```
LINK object-list,runfile,listfile,lib-list[/x...]
```

where ***object-list*** is a list of object modules, separated by plus signs;
runfile is the name of the file to receive the executable output;
listfile is the name of the file to receive the listing;
lib-list is a list of library modules to be searched; and
/***x*** are the optional switches, further described under the Switches section later in this chapter, which may be placed following any of the response entries (just before any of the commas or after the *lib-list*, as shown).

The entries following LINK are responses to the command prompts. The entry fields for the different prompts must be separated by commas. For each file name specified, you can optionally specify the drive name and path name.

To select the default for a field, simply enter a second comma without spaces in between:

```
LINK FUN+TEXT+TABLE+CARE/P/M,,FUNLIST,COBLIB.LIB
```

This example causes LINK to be loaded, then causes the object modules FUN.OBJ, TEXT.OBJ, TABLE.OBJ, and CARE.OBJ to be loaded. When LINK is ready to produce the .EXE run file, it pauses (because of the /P switch). When you press ENTER, LINK produces the .EXE run file, produces a global symbol map (because of the /M switch), defaults to FUN.EXE run file, creates a list file named FUNLIST.MAP, and searches the library file COBLIB.LIB.

Response File Method

The Response File Method enables you to enter all of the specifications for a LINK operation in a single file.

```
LINK @filespec
```

where @ is a pointer that tells LINK that all the prompt answers are contained in the file whose name follows; and

filespec is the name of a response file with an optional drive name and/or path name. A response file contains answers to the LINK prompts (summarized under the Command Prompt Method for invoking), and may also contain any of the switches. The Response File Method permits the user responses to the LINK prompts.

NOTE: Before using the Reponse File Method to invoke LINK, the user must first create the response file. You can create a response file using EDLIN, the MS-DOS editor, the COPY command or a text editor.

A response file has text lines, one for each prompt. Responses must appear in the same order as the command prompts appear.

Use switches and command characters in the response file the same way as they are used for responses entered on the terminal keyboard.

When the LINK session begins, each prompt will be displayed in turn with the responses from the response file. If the response file does not contain answers for all the prompts, (either in the form of file names or the semicolon command character or RETURNs) LINK will, after displaying the prompt which does not have a response, wait for you to enter a legal response. When a legal response has been entered, LINK continues the link session.

The response file

```
FUN TEXT TABLE CARE
/PAUSE/MAP
FUNLIST
COBLIB.LIB
```

causes LINK to load the four object modules. LINK will pause before producing the .EXE run file to permit you to swap disks (see discussion under /PAUSE in Switches section before using this feature). After you press ENTER, LINK creates a PUBLIC symbol map. The output files will be named FUN.EXE and FUNLIST.MAP; LINK will search the library file COBLIB.LIB, and will use the default settings for the flags.

Command Prompts

LINK is operated by entering responses to four text prompts. When you have entered a response to the current prompt, the next appears. When the last prompt has been answered, LINK begins linking automatically without further command. When the link session is successfully finished, LINK exits to the operating system. If the link session is unsuccessful, LINK returns the appropriate error message.

LINK prompts you for the names of object, run, and list files; and for libraries. The prompts are listed here in their order of appearance. For prompts which can default to preset responses, the default response is shown in square brackets ([]) following the prompt. The `Object Modules:` prompt is followed by only a file name extension default response because it has no preset file name response and requires a file name from you to be entered.

Object Modules [.OBJ]:

Enter a list of the object modules to be linked. LINK assumes by default that the file name extension is .OBJ. If an object module has any other file name extension, the extension must be given here. Otherwise, the extension may be omitted.

Modules must be separated by plus signs (+).

Remember that LINK loads segments into classes in the order encountered (see the Definitions section at the beginning of this chapter). Use this information for setting the order in which the object modules are entered.

Run File [Objfile.EXE]:

The file name entered will be created to store the run (executable) file that results from the link session. All run files receive the file name extension .EXE, even if you specify an extension (in other words, your specified extension is ignored).

If no response is entered to the `Run File [Objfile.EXE]:` prompt, LINK uses the first file name entered in response to the Object Modules prompt as the run file name. For example, if you enter

```
Run File [FUN.EXE]: B:PAYROLL/P
```

LINK is directed to create the run file PAYROLL.EXE on drive B. Also, LINK will pause, which allows you to insert a new disk to receive the run file.

List File [NUL.MAP]:

The list file contains an entry for each segment in the input (object) modules. Each entry also shows the offset (addressing) in the run file.

The default response is that no (NUL) list file is created with the .MAP extension.

Libraries [.LIB]:

The valid responses are one to ten library file names or pressing RETURN. (A RETURN only means no library search.) Library files must have been created by a library utility. LINK assumes by default that the file name extension is .LIB for library files.

Library file names must be separated by plus signs (+).

LINK searches the library files in the order listed to resolve external references. When it finds the module that defines the external symbol, LINK processes the module as another object module.

If LINK cannot find a library file on the disk in the disk drives, it returns the message

```
Cannot find library library-name
Enter new drive letter:
```

Simply press the letter for the drive name (for example, press **B**).

LINK does not search within each library file sequentially. LINK uses a method called *dictionary indexed library search*. This means that LINK finds definitions for external references by index access, rather than searching from the beginning of the file to the end for each reference. This indexed search reduces substantially the link time for any sessions involving library searches.

Switches

The six switches control alternate linker functions. Switches must be entered at the end of a prompt response, regardless of which method is used to invoke LINK. Switches may be grouped at the end of any one of the responses, or may be scattered at the end of several. Each switch must be preceded by the slash mark (/).

All switches may be abbreviated. You can enter anything from a single letter through the whole switch name. The only restriction is that an abbreviation must be a sequential sub-string from the first letter through the last entered; no gaps or transpositions are allowed. For example:

LEGAL	ILLEGAL
/D	/DSL
/DS	/DAL
/DSA	/DLC
/DSALLOCA	/DSALLOCT

/DSALLOCATE

Use of the /DSALLOCATE switch directs LINK to load all data (DGroup) at the high end of the Data Segment. Otherwise, LINK loads all data at the low end of the Data Segment. At runtime, the DS pointer is set to the lowest possible address and allows the entire DS segment to be used. Use of the /DSALLOCATE switch in combination with the default load low, (that is, the /HIGH switch is not used) permits your application program to allocate dynamically any available memory below the area specifically allocated within DGroup, yet to remain addressable by the same DS pointer. This dynamic allocation is needed for Pascal and FORTRAN programs.

NOTE: Your application program may dynamically allocate up to 64K bytes (or the actual amount available) less the amount allocated within DGroup.

/HIGH

Use of the /HIGH switch causes LINK to place the run image as high as possible in memory. Otherwise, LINK places the run file as low as possible.

NOTE: Do not use the /HIGH switch with Pascal or FORTRAN programs.

/LINENUMBERS

Use of the /LINENUMBERS switch directs LINK to include in the list file the line numbers and addresses of the source statements in the input modules. Otherwise, line numbers are not included in the list file.

NOTE: Not all compilers produce object modules that contain line number information. In these cases, of course, LINK cannot include line numbers.

/MAP

/MAP directs LINK to list all PUBLIC (global) symbols defined in the input modules. If /MAP is not given, LINK will list only segment definitions and errors (which includes undefined globals).

The symbols are listed alphabetically. For each symbol, LINK lists its value and its segment:offset location in the run file. The symbols are listed at the end of the list file.

/PAUSE

The /PAUSE switch causes LINK to pause in the link session when the linker is about to produce the .EXE run file. Normally, LINK performs the linking session without stop from beginning to end. This allows the user to swap the disks before LINK outputs the run (.EXE) file.

When LINK is ready to produce the run file, it displays the message:

```
About to generate .EXE file
Change disks <hit ENTER>
```

LINK resumes processing after you press ENTER.

CAUTION: Do not swap the disk which is to receive the list file, or the disk used for the VM.TMP file, if created.

/STACK:number

number represents any positive numeric value (in hexadecimal radix) up to 65536 bytes. If the /STACK switch is not used for a link session, LINK calculates the necessary stack size automatically.

If a value from 1 to 511 is entered, LINK uses 512.

All compilers and assemblers should provide information in the object modules that allow the linker to compute the required stack size.

/NO

This switch tells LINK to not search the default (product) libraries in the object modules. For example, if you are linking object modules to Pascal, specifying the /NO switch tells LINK to not automatically search the library named PASCAL.LIB to resolve external references.

Error Messages

All errors, except No STACK segment, cause the link session to abort. Therefore, after the cause is found and corrected, LINK must be rerun.

```
Attempt to access data outside of segment bounds, possibly bad ob-
ject module
```

EXPLANATION: This error occurs because you have a bad object file. Correct the program, reassemble your module and rerun LINK.

```
Bad numeric parameter
```

EXPLANATION: The numeric value is not in digits. Change the numeric value and rerun LINK.

`Cannot find file` *`filename`*

EXPLANATION: The file specified does not exist on this disk. Change the file name or insert the disk on which the file resides.

```
Cannot open temporary file
```

EXPLANATION: LINK is unable to create the file VM.TMP because the disk directory is full. Try another disk that contains more space, or free up some space on the disk by deleting files.

```
Error: Dup record too complex
```

EXPLANATION: The DUP record in the Assembly language module is too complex. Simplify the DUP record, reassemble and rerun LINK.

```
Error: Fixup offset exceeds field width
```

EXPLANATION: An Assembly language instruction refers to an address with a short instruction where a long instruction is required. Edit the Assembly language source and reassemble.

LINK

Error Messages

```
Input file read error
```

EXPLANATION: This error occurs when there is a bad object file (probably a bad disk). Reassemble the module and rerun LINK.

```
Invalid object module
```

EXPLANATION: This error occurs when the object module(s) is incorrectly formed or incomplete (as when assembly was stopped in the middle). Reassemble the module and rerun LINK.

```
Program size or number of segments exceeds capacity of Linker
```

EXPLANATION: The total size may not exceed 900K bytes and the number of segments and classes taken together may not exceed 255. Change your program to fit within these boundaries and rerun LINK.

```
Requested stack size exceeds 64K
```

EXPLANATION: 64K bytes is the stack size limit. Rerun LINK specifying a size less than 64K bytes with the /STACK switch.

```
Segment size exceeds 64K
```

EXPLANATION: 64K bytes is the addressing system limit. Edit the Assembly language file to make the segment less than 64K, reassemble and rerun LINK.

```
Symbol defined more than once
```

EXPLANATION: LINK found two or more modules that define a single symbol name. Edit one of the assembly language modules, reassemble and rerun LINK.

```
Symbol table capacity exceeded
```

EXPLANATION: Very many, and/or very lengthy names were entered; exceeding approximately 25K bytes. Reedit the source files by using shorter names for variables, reassemble and rerun LINK.

`Too many external symbols in one module`

EXPLANATION: The number of external symbols per module exceeds the limit of 1024. Edit the Assembly language file, splitting the externals so each module is within the 1024 limit. Reassemble and rerun LINK.

`Too many groups`

EXPLANATION: The number of groups exceeds the limit of 10. Edit the source modules to use 10 or less groups. Reassemble and rerun LINK.

`Too many libraries specified`

EXPLANATION: The number of libraries exceeds the limit of 10. Rerun LINK specifying 10 or fewer libraries (Consolidate libraries if necessary).

`Too many PUBLIC symbols`

EXPLANATION: The number of PUBLIC symbols exceeds the limit of 1024. Reedit one or more Assembly language files defining fewer than 1024 PUBLIC symbols. Rerun LINK.

`Too many segments or classes`

EXPLANATION: The number of segments or classes exceeds the limit of 255 (segments and classes taken together). Reedit one or more Assembly language files defining fewer than 255 segments and/or classes. Rerun LINK.

`Unresolved externals:` *`list`*

EXPLANATION: *list* represents the external symbols which do not have a defining module defining them among the modules or library files specified. Reexamine the use of PUBLIC and external symbols in your Assembly language programs. Make the necessary changes and rerun LINK.

LINK

Error Messages

`VM read error`

EXPLANATION: This error occurs because of a disk problem and is not caused by LINK. Use a different disk.

`Warning: No STACK segment`

EXPLANATION: None of the object modules specified contains a statement that allocates stack space, but you entered the /STACK switch. Invoke LINK without the /STACK switch or reedit the module creating a stack segment, reassemble and rerun LINK.

NOTE: When creating a .COM file using EXE2BIN, or if the program allocates its own stack, this warning can be ignored.

`Warning: Segment of absolute or unknown type`

EXPLANATION: This error occurs because there is a bad object module or an attempt was made to link modules that LINK cannot handle (such as an absolute object module). Reassemble the offending module and rerun LINK.

`Write error in TMP file`

EXPLANATION: This error occurs because the disk does not have space remaining to expand VM.TMP file. Try another disk that contains more space, or free up some space on the disk, and rerun LINK.

`Write Error On Run File`

EXPLANATION: This error occurs because the disk does not have enough space for the run file. Try another disk that contains more space, or free up some space on the disk, and rerun LINK. (The /P switch could be used to solve this problem.)

Chapter 15
DEBUG

Purpose

DEBUG is an MS-DOS utility which provides a controlled testing environment for isolating and eliminating errors in, or malfunctions of, binary and executable object files. It is a tool for software development that provides a controlled environment to load, examine, and change a program.

NOTE: This chapter assumes that the user is familiar with Assembly language programming and with the operation of 8086 and 8088 registers and instructions. The user should also know the definition of microcomputer terms such as: bit, byte, flag, and register.

Entry Form

```
DEBUG [filespec [arglist]] RETURN
```

where **DEBUG** invokes the DEBUG command and causes the DEBUG prompt to appear;
filespec is the file to be debugged; and
arglist is a listing of file name functions and parameters that are applied to the program *filespec*.

Preliminary Concepts

Just as a text editor (like EDLIN) is used to alter text files, DEBUG alters binary files. DEBUG often eliminates the need to reassemble a program to see if a problem has been corrected by a minor change. It allows you to alter the contents of a file or the contents of a CPU register and then to immediately reexecute a program to check on the validity of the changes.

All DEBUG functions (summarized in Table 15.1) may be aborted at any time by holding down the CTRL key and pressing the BREAK key (CTRL-BREAK). DEBUG CTRL-NUM LCK suspends displays so that you can read parts of them before they scroll away. Making any entry other than CTRL-BREAK, CTRL-C, CTRL-S, or CTRL-NUM LCK restarts the display. All of these entries are consistent with the control character functions available at the MS-DOS command level.

When DEBUG is invoked, it sets up a program header at offset 0 in its program work area. In previous versions, it was acceptable to overwrite this header with impunity: this is true of the default header setup if no *filespec* is given to DEBUG. If debugging a .COM or .EXE file, however, you must be careful not to tamper with the header of the program below address 5CH. To do this will probably result in a crash. It is also important that no attempt is made to "restart" a program once the `Program terminated normally` message is given. The program must be reloaded with the N (Name) and L (Load) functions in order for it to run properly.

Invoking DEBUG

To invoke DEBUG you must enter a command at the system prompt in the following form:

```
DEBUG [filespec [arglist]]
```

where ***filespec*** is the file specification for the file to be debugged; and

arglist is any functions and/or parameters to be applied to *filespec*.

For example, if just a *filespec* is specified, as shown:

```
DEBUG FILE.EXT
```

DEBUG loads the *filespec* FILE.EXT into memory in the lowest available segment. The BX:CX registers are loaded with the number of bytes placed into memory. The DEBUG prompt is a hyphen (-).

An *arglist* may be specified if *filespec* is present. These are file name parameters and switches that are to be passed to the program *filespec*. Thus, when *filespec* is loaded into memory, it is loaded as if it had been invoked with the command:

```
filespec arglist
```

Here, *filespec* is the file to be debugged, and the *arglist* is the rest of the command line, consisting of functions and parameters, that is used when *filespec* is invoked and loaded into memory.

If you do not wish to specify a *filespec*, then enter your DEBUG command as follows:

```
DEBUG
```

and press **RETURN**.

DEBUG returns with the – (hyphen) prompt, signaling that it is ready to accept your functions. Since no file name has been specified, current memory, disk sectors, or disk files can be worked on by invoking subsequent functions.

Functions

Each DEBUG function consists of a single letter followed by one or more parameters. Additionally, the control character and special editing functions described in Chapter 6, "Command Functions," all apply in the DEBUG program operation.

If a syntax error occurs in a DEBUG function, DEBUG reprints the function command line and indicates the error with a ^ (carat) and the word `Error`.

For example:

```
DCS:100 CS:110
     ^ Error
```

All functions and parameters may be entered in either upper- or lowercase. Any combination of upper- and lowercase may be used in command lines.

The DEBUG functions are summarized in Table 15.1. They are described in detail, with examples, following the description of function parameters.

Table 15.1. Summary of DEBUG Functions

FUNCTION SYNTAX	FUNCTION NAME
A[*address*]	Assemble
C*range address*	Compare
D[*address*][L*value*] D[*range*]	Dump
E*address*[*list*]	Enter
F*range list*	Fill
G[=*address1*][*address2*...]	Go
H*address address*	Hex
I*value*	Input
L[*address*[*drive record record*]]	Load
M*range address*	Move
N*filespec*[*filespec*...]	Name
O*value byte*	Output
Q	Quit
R[*register*]	Register
S*range list*	Search
T[=*address*][*value*]	Trace
U[*address*][L *value*] U[*range*]	Unassemble
W[*address*[*drive record record*]]	Write

Function Parameters

Table 15.1 shows that all DEBUG functions, with the exception of the Quit function, accept parameters. Parameters may be separated by delimiters (spaces or commas), but a delimiter is required only between two consecutive hexadecimal values. Thus, the following functions

```
DCS:100 110
```

and

```
D CS:100 110
```

are equivalent. (Remember, entries may be made in any combination of upper- or lowercase.)

Preliminary Concepts

PARAMETER	DEFINITION

address	A two part designation consisting of one of the following:
	• an alphabetic segment register designation plus an offset value, or
	• a four-digit segment address plus an offset value.
	The segment designation or segment address may be omitted, in which case the default segment is used. DS is the default segment for all commands except A, G, L, T, U, and W, for which the default segment is CS. All numeric values are hexadecimal. For example,
	`CS:0100`
	`04BA:0100`
	The colon is required between a segment designation (whether numeric or alphabetic) and an offset.

byte	A two-digit hexadecimal value to be placed in or read from an address or register.
drive	A one-digit hexadecimal value to indicate which drive a file will be loaded from or written to. These values designate the drives, for example: 0=A:, 1=B:, 2=C:, 3=D:, 4=E:, 5=F:, 6=G:, and 7=H:.

list — A series of *byte* values or a *string*. *List* must be the last parameter on the command line, as in

```
FCS:100 42 45 52 54 41
```

range — Either two *addresses*: such as *address address*; or one *address*, an L, and a *value*: such as *address* L *value*, where *value* is the length (number of bytes) the function should operate on. For example,

```
CS:100 110
```
or
```
CS:100 L 10
```

are valid, while the following example

```
2000:100 4000:110
           ↑ Error
```

is invalid. The limit for *range* is 10000 hex. To specify a *value* of 10000 hex within four digits, enter 0000 (or 0).

record — A one- to four-digit hexadecimal value used to indicate the logical record number on the disk and the number of disk sectors to be written or loaded. Logical records correspond to sectors. However, their numbering differs since they represent the entire disk space.

register — A one- to two-digit alphabetic designation for an area in memory used to hold intermediate results, such as operands for arithmetic and logical operations, offset addresses within segments, and specifying starting addresses of segments.

DEBUG

Preliminary Concepts

string

Any number of characters enclosed in quote marks. Quote marks may be either single (') or double("). Within *strings*, the opposite set of quote marks may be used freely as literals. If the delimiter quote marks must appear within a *string*, the quote marks must be doubled. For example, the following strings are valid:

'This as a “string” is okay.'
or
'This as a ''string'' is okay.'

However, the following string is invalid:

'This as a 'string' is not okay.'

Similarly, these strings are valid:

“This as a 'string' is okay.”
“This as a ““string”” is okay.”

However, this string is invalid:

“This as a “string” is not okay.”

Note that the double quotations are not necessary in the following strings:

“This as a ' 'string' ' is not necessary.”
and
'This as a “ “string” ” is not necessary.'

The ASCII values of the characters in the string are used as a *list* of byte values.

value

A one- to four-digit hexadecimal value used to specify a port number or the number of times a command should repeat its functions.

DEBUG Function Descriptions

The following pages describe the syntax for the DEBUG functions, their parameters, purpose, and the special conditions that apply to some functions.

Assemble

Syntax

A[*address*]

Purpose

To assemble 8086/8087/8088 mnemonics directly into memory.

Comments

All numeric values are hexadecimal and may be entered as 1-4 characters. If a syntax error is encountered, DEBUG responds with

```
^ Error
```

and redisplays the current assembly address.

DEBUG

Assemble

Prefix mnemonics must be entered in front of the opcode (operation code) to which they refer. They may be entered on the same line, or on a separate preceding line. The prefix mnemonics consist of

REP	repeat
REPZ	repeat while zero
REPE	repeat while equal
REPNZ	repeat while not zero
REPNE	repeat while not equal

The segment override prefix mnemonics are CS:, DS:, ES:, and SS:. These mnemonics *must* be entered on a separate line immediately preceding the segment override.

For string manipulation mnemonics, you must explicitly state the string size. For example, the MOVSW must be used to move word strings and MOVSB must be used to move byte strings.

The mnemonic for the far return is RETF.

The assembler will automatically assemble short, near or far jumps and calls, depending on byte displacement (quantity representing change of position) to the destination address. These may be overridden with the NEAR or FAR prefix. The NEAR and FAR prefixes are used to designate the location of a call or jump. For example:

```
0100:0500 JMP   502         ; a 2 byte short jump
0100:0502 JMP   NEAR 505    ; a 3 byte near jump
0100:0505 JMP   FAR 50A     ; a 5 byte far jump
```

The NEAR prefix may be abbreviated to NE but the FAR prefix cannot be abbreviated.

DEBUG cannot tell whether some operands refer to a word memory location or a byte memory location. Therefore, the data type must be explicitly stated with the prefix "WORD PTR" or "BYTE PTR". DEBUG will also accept the abbreviations "WO" and "BY". For example:

```
Neg    BYTE PTR [128]
DEC    WO PTR [SI]
```

In addition, DEBUG cannot tell whether an operand refers to a memory location or to an immediate operand. DEBUG uses the common convention that operands enclosed in square brackets refer to memory. For example:

```
MOV    AX,21            ;Load AX with 21H
MOV    AX,[21]          ;Load AX with the contents
                        ;of memory location 21H
```

Two popular pseudo operations have also been included. The DB opcode will assemble byte values directly into memory. The DW opcode will assemble word values directly into memory. For example:

```
DB     1,2,3,4,"TH1S 1S AN EXAMPLE"
DB     'THIS IS A QUOTE: "'
DB     "THIS 1S A QUOTE: '"

DW     1000,2000,3000,"BACH"
```

All forms of the register indirect commands are supported. For example

```
ADD    BX,34[BP+2].[S1-1]
POP    [BP+D1]
PUSH   [S1]
```

All opcode synonyms are supported. For example

```
LOOPZ  100
LOOPE  100
JA     200
JNBE   200
```

For 8087 opcodes the WAIT or FWAIT prefix must be explicitly specified. For example

```
FWAIT  FADD  ST,ST(3)    ; Tbis line will assemble
                         ; a FWAIT prefix

FLD TBYTE PTR [BX]       ; This line will not
```

Compare

Syntax

```
Crange address
```

Purpose

Match the portion of memory specified by *range* to a portion of the same size beginning at *address*.

Comments

If the two areas of memory are identical, there is no display and DEBUG returns with the MS-DOS system prompt. If there are differences, they are displayed as:

```
address1 byte1 byte2 address2
```

The following functions have the same effect:

```
C100,1FF 300
```

or

```
C100L101 300
```

Each function compares the block of memory from 100 to 1FFH with the block of memory from 300 to 3FFH.

Dump

Syntax

```
D[address][ L value]
D[range]
```

Purpose

Beginning at the *address* specified, display the memory contents of either a single address, a range of addresses, or the number of lines specified by *value*.

Comments

If only a single address is specified, the contents of 128 bytes are displayed. If a range of addresses is specified, the contents of the range are displayed. If the D function is entered without parameters, the result is the same as if you specified a single address, except that the display begins at the current location in the DS (data segment).

The dump is displayed in three portions: the address (the segment register address and the offset), a hexadecimal dump (each byte is shown in hexadecimal value), and an ASCII dump (the bytes are shown in ASCII characters). Nonprinting characters are denoted by a period (.) in the ASCII portion of the display. Each display line shows sixteen bytes with a hyphen between the eighth and ninth bytes.

Each displayed line begins on a 16-byte boundary. If the initial address is not on a 16-byte boundary, the first and last line may contain fewer bytes than the rest of the displayed lines (but together, the incomplete first and last line equal a complete display line—128 bytes).

Dump

Assume the following function is entered:

```
DCS:100 10F
```

DEBUG displays

```
04BA:0100 42 45 52 54 41 20 54 00-20 42 4F 52 4C 41 4E 44 BERTA T.  BORLAND
```

If just the D function is entered, the display is formatted as described above. Each line of the display begins with an address; on an even 16-byte boundary. Each subsequent D (entered without parameters) displays the bytes immediately following those last displayed.

If you enter the function

```
DCS:100 L 20
```

the display is formatted as described above, but 20H bytes are displayed.

If you enter the function

```
DCS:100 115
```

the display is formatted as described above, but all the bytes in the range of addresses from 100H through 115H in the CS segment are displayed.

Enter

Syntax

`Eaddress[ list]`

Purpose

Enter an *address*, display its contents, and wait for input.

Comments

If the optional *list* of hexadecimal values is entered, the replacement of byte values occurs automatically. (If an error occurs, no byte values are changed.)

If the *address* is entered without the optional list, DEBUG displays the address and its contents, then repeats the address on the next line and waits for your input.

While DEBUG waits for your input, you can perform any of the following actions:

1. Replace a byte value with a value you type in. Simply type the value after the current value.

   If the value you enter is not a valid hexadecimal value or if more than two digits are typed, the invalid or extra character is not echoed.

Enter

2. Press the SPACE BAR to advance to the next byte.

   To change the value, simply enter the new value (as described in step 1). If you enter spaces to advance beyond an eight-byte boundary, DEBUG starts a new display line with the address displayed at the beginning.

3. Type a hyphen (–) to return to the preceding byte.

   If you decide to change a byte that precedes the current position, typing the hyphen returns the current position to the previous byte. When the hyphen is typed, a new line is started with the address and its byte value displayed.

4. Press the **RETURN** key to terminate the Enter function. The RETURN key may be pressed at any byte position.

Assume the following function is entered:

```
ECS:100
```

DEBUG displays

```
04BA:0100 EB.
```

To change this byte value to 41, enter "41" as shown:

```
04BA:0100   EB.41
```

To step through the subsequent bytes, press the SPACE BAR to see:

```
04BA:0100   EB.41     10.     00.     BC.
```

To change BC to 42, enter "42" as shown:

```
04BA:0100   EB.41     10.     00.     BC.42
```

Now, realizing that 10 should be 6F, enter the hyphen as many times as needed to return to byte 0101 (value 10). Then replace 10 with 6F.

```
04BA:0100  EB.41    10.    00.    BC.42
04BA:0102  00.-
04BA:0101  10.6F
```

Pressing the RETURN key ends the Enter function and returns the DEBUG (-) hyphen prompt.

Fill

Syntax

`Frange list`

Purpose

Fill the addresses in the *range* with the values in the *list*.

Comments

If the *range* contains more bytes than the number of values in the *list*, the *list* will be used repeatedly until all bytes in the *range* are filled. If the *list* contains more values than the number of bytes in the *range*, the extra values in the *list* will be ignored. If any of the memory in the *range* is not valid (bad or nonexistent), the error will be propagated into all succeeding locations.

The F function does not abort as the E function does. Yet, the F function is a multiple version of the E function in that it allows you to change more than one address at a time.

Assume the following function is entered:

```
F04BA:100 L 100 42 45 52 54 41
```

DEBUG fills memory locations 04BA:100 through 04BA:1FF with the bytes specified. The five values are repeated until all 100H bytes are filled.

Go

Syntax

```
G[=address1][ address2...]
```

Purpose

Execute the program currently in memory.

Comments

If the Go function is entered alone, the program executes as if the program had run outside DEBUG.

If *=address1* is set, execution begins at the address specified. If the segment designation is omitted from *=address1*, only the instruction pointer is set. If the segment designation is included in *=address1*, both the CS segment and the instruction pointer are set. The equal sign (=) is required, so that DEBUG can distinguish the start *=address1* from the breakpoint addresses (*address2...*).

If you have other optional addresses set, execution will stop at the first *address* encountered (regardless of that address' position in the list of addresses to halt execution) no matter which branch the program takes. When program execution reaches a breakpoint, the registers, flags, and decoded instruction are displayed for the next instruction to be executed. (The result is the same as if you had entered the Register function for the breakpoint address.)

Up to 10 breakpoints may be set. Breakpoints may be set only at addresses containing the first byte of an 8088 opcode. If more than 10 breakpoints are set, DEBUG returns the `BP` error message.

Your stack pointer must be valid and have six bytes available for this function. The G function uses an IRET (Return from Interrupt) instruction to cause a jump to the program under test. Your stack pointer is set, and your flags, code segment register, and instruction pointer are pushed on your stack. (Thus, if your stack is not valid or is too small, the operating system may crash.)

An interrupt code (0CCH) is placed at the specified breakpoint address(es). When an instruction with the breakpoint code is encountered, all breakpoint addresses are restored to their original instructions. If execution is not halted at one of the breakpoints, the interrupt codes are not replaced with the original instructions.

For example, assume the following function is entered:

```
GCS:7550
```

The program currently in memory executes up to the address 7550 in the CS segment. Then DEBUG displays registers and flags, after which the Go function is terminated.

After a breakpoint has been encountered, if you enter the Go function again, then the program executes just as if you had entered the file name at the MS-DOS command level. The only difference is that program execution begins at the instruction at the breakpoint address rather than at the usual start address.

Hex

Syntax

```
Haddress address
```

Purpose

Perform hexadecimal arithmetic on the two parameters.

Comments

First, DEBUG adds the two parameters, then subtracts the second parameter from the first. The results of the arithmetic are displayed on one line; first the sum, then the difference.

For example, assume the following function is entered:

```
H10A 19F
```

DEBUG performs the calculations and then returns the results:

```
02A9  FF6B
```

Input

Syntax

```
Ivalue
```

Purpose

Input and display one byte from the port specified by value.

Comments

A 16-bit port address is allowed.

Assume the following function is entered:

```
I2F8
```

Assume also that the byte at the port is 42H. DEBUG inputs the byte and displays the value:

```
42
```

Load

Syntax

`L[address[ drive record record]]`

Purpose

Load a file into memory.

Comments

Sets BX:CX to the number of bytes read. The file must have been named either with the DEBUG invocation command or with the N function. Both the invocation and the N functions format a file name properly in the normal format of a file control block at DS:5C.

If the L function is given without any parameters, DEBUG loads the file into memory (unless it's a .EXE file) beginning at address CS:100 and sets BX:CX to the number of bytes loaded.

If the L function is given with an address parameter, loading begins at the memory *address* specified.

If L is entered with all parameters, absolute disk sectors (rather than a file) are loaded. The *records* are taken from the *drive* specified, (where drive names are represented by numbers—0=A:, 1=B:, 2=C:, 3=D:, 4=E:, 5=F:, 6=G:, and 7=H:). DEBUG begins loading with the first *record* specified; and continues until the number of sectors specified in the second *record* has been loaded.

Assume the following entries are made (the first at the system prompt and the second at the DEBUG prompt):

```
B>A:DEBUG
-NFILE.COM
```

Now, to load FILE.COM, enter:

```
L
```

DEBUG loads the file and returns the DEBUG prompt. If you want to load only portions of a file or certain records from a disk, enter

```
L04BA:100 2 0F 6D
```

DEBUG then loads 109 (6D hex) records from drive C (2), beginning with logical record number 15, into memory beginning at address 04BA:0100. When the records have been loaded, DEBUG simply returns the (-) prompt.

If the file has a .EXE extension, it is relocated to the load *address* specified in the header of the .EXE file; the address parameter is always ignored for .EXE files. Note that the header itself is stripped from the .EXE file before it is loaded into memory. Thus, the size of a .EXE file on disk will differ from its size in memory.

Move

Syntax

Mrange address

Purpose

Move the block of memory specified by *range* to the location beginning at the *address* specified.

Comments

Overlapping moves (moves where part of the block overlaps some of the current addresses) are always performed without loss of data.

Addresses that could be overwritten are moved first. The sequence for moves from higher addresses to lower addresses is to move the data beginning at the block's lowest address and work towards the highest. The sequence for moves from lower addresses to higher addresses is to move the data beginning at the block's highest address and work towards the lowest.

NOTE: If the addresses in the block being moved will not have new data written to them, the data that was there before the move will remain. The M function really copies the data from one area into another, in the sequence described, and writes over the new addresses. That is why the sequence of the move is important.

For example, if you enter

```
MCS:100 110 CS:500
```

DEBUG first moves address CS:110 to address CS:510, then CS:10F to CS:50F, and so on until CS:100 is moved to CS:500. You should enter the D function, using the *address* entered for the M function, to review the results of the move.

Name

Syntax

```
Nfilespec[ filespec...]
```

Purpose

Set file names.

Comments

The Name function performs two distinct actions, both having to do with file names. First, Name is used to assign a file name, with an optional drive name and/or path name, for a later Load or Write function. Thus, if you invoke DEBUG without naming any file to be debugged, the N*filespec* function must be entered before a file can be loaded. Second, Name is used to assign file name parameters to the file being debugged. In this case, Name accepts a list of parameters that are used by the file being debugged.

Name

These functions overlap. Consider the following set of DEBUG functions:

```
-NFILE1.EXE
-L
-G
```

Because of the two-pronged effect of the Name function, the following happens:

1. (N)ame assigns the name FILE1.EXE to the file to be used in any later Load or Write functions.

2. (N)ame also assigns the file name FILE.EXE to the first file name parameter to be used by any program that is later debugged.

3. (L)oad loads FILE.EXE into memory.

4. (G)o causes FILE.EXE to be executed with FILE.EXE as the single file name parameter (that is, FILE.EXE is executed as if the file FILE.EXE had been typed at the command level.)

A more useful chain of functions might go like this:

```
-NFILE1.EXE
-L
-NFILE2.DAT FILE3.DAT
-G
```

In this example, Name sets FILE1.EXE as the file name for the subsequent Load function. The Load function loads FILE1.EXE into memory, and then the Name function is used again—this time to specify the parameters to be used by FILE1.EXE. Finally, when the Go function is executed, FILE1.EXE is executed as if FILE1 FILE2.DAT FILE3.DAT had been typed at the MS-DOS command level.

Name

NOTE: If a Write function were executed at this point, then FILE1.EXE—the file being debugged—would be saved with the name FILE2.DAT! To avoid such undesired results, you should always execute a Name function before either a Load function or a Write function.

There are four distinct regions of memory that can be affected by the Name function:

CS:5C	FCB for file 1
CS:6C	FCB for file 2
CS:80	Count of characters
CS:81	All characters entered

A File Control Block (FCB) for the first file name parameter given to the Name function is set up at CS:5C. If a second file name parameter is given, then an FCB is set up for it beginning at CS:6C. The number of characters typed in the Name function (exclusive of the first character, "N") is given at location CS:80. The actual stream of characters given by the Name function (again, exclusive of the letter "N") begins at CS:81. Note that this stream of characters may contain switches and delimiters that would be valid in any command typed at the MS-DOS system prompt.

A typical use of the Name function would be:

```
A>DEBUG PROG.COM
-NPARAM1 PARAM2/C
-G
-
```

In this case, the Go function executes the file in memory as if the following command line had been entered at the MS-DOS system prompt:

```
PROG PARAM1 PARAM2/C
```

Testing and debugging, therefore, reflect a normal runtime environment for PROG.COM.

Output

Syntax

```
Ovalue byte
```

Purpose

Send the *byte* specified to the output port specified by *value*.

Comments

A 16-bit port address is allowed.

If you enter

```
02F8 4F
```

and press **RETURN**, DEBUG outputs the byte value 4F to output port 2F8.

Quit

Syntax

```
Q
```

Purpose

Terminate the debugger.

Comments

The Q function takes no parameters and exits DEBUG without saving the file currently being operated on. You are returned to the MS-DOS command level.

To end the debugging session, enter:

```
Q
```

and press **RETURN**. DEBUG is terminated and control returns to the MS-DOS command level.

Register

Syntax

`R[register]`

Purpose

Display the contents of one or more CPU registers.

Comments

If no *register* is entered, the R function dumps the register save area and displays the contents of all registers and flags.

If a register name is entered, the 16-bit value of that register is displayed in hexadecimal, and then a colon appears as a prompt. Then either enter a *value* to change the register or simply presses the RETURN key if no change is wanted.

The only valid *registers* are:

AX BP SS
BX SI CS
CX DI IP (IP and PC both refer to the
DX DS PC instruction pointer.)
SP ES F

Any other entry for *register* will result in the `BR` error message.

Table 15.2. Register Flag Codes

FLAG NAME	SET		CLEAR	
Overflow	OV	Overflow	NV	No overflow
Direction	DN	Decrement	UP	Increment
Interrupt	EI	Enabled	DI	Disabled
Sign	NG	Negative	PL	Plus
Zero	ZR	Zero	NZ	Not zero
Auxiliary carry	AC	Aux. carry	NA	No aux. carry
Parity	PE	Even	PO	Odd
Carry	CY	Carry	NC	No carry

If F is entered as the *register*, DEBUG displays a list of two character alphabetic codes. To alter a flag, enter the alternate two letter code. The flags are either *set* or *clear*.

The flags, with their codes for set and clear, are listed in Table 15.2.

Whenever you enter the function RF, the flags are displayed in the order shown above in a row at the beginning of a line. At the end of the list of flags, DEBUG displays a static hyphen (-). You may enter new flag values as alphabetic pairs. The new flag values can be entered in any order. You are not required to leave spaces between the flag entries. To exit the R function, press the **RETURN** key. The DEBUG prompt (–) appears. Flags for which new values were not entered remain unchanged.

If more than one value is entered for a flag, DEBUG returns a `DF` error message. If you enter a flag code other than those shown above, DEBUG returns a `BF` error message. In both cases, the flags up to the error in the list are changed; flags at and after the error are not.

Register

At startup, the segment registers are set to the bottom of free memory, the instruction pointer is set to 0100H, the stack pointer is set to 005AH, all flags are cleared, and the remaining registers are set to zero.

When you enter

```
R
```

and press **RETURN**, DEBUG displays all registers, flags, and the decoded instruction for the current location. If the location is CS:11A, then DEBUG might display

```
AX=0000  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0BCB  SI=0B0B  DI=0000
0S=0CCA  ES=0CCA  SS=0CCA  CS=0CCA  IP=0100   NV UP EI PL NZ NA PO NC
0CCA:0100 0000          ADD      [BX+SI],AL                      DS:0B0B=5A
```

If you enter

```
RF
```

and press the **RETURN** key, DEBUG displays the flags:

```
NV UP DI NG NZ AC PE NC -
```

Now, enter any valid flag designation, in any order, with or without spaces. For example, enter

```
NV UP DI NG NZ AC PE NC - PLEICY
```

and press the **RETURN** key. DEBUG responds only with the DEBUG prompt. To see the changes, you can enter either the R, or RF function. For example, enter

```
RF
```

and press the **RETURN** key. The screen displays

```
NV UP EI PL NZ AC PE CY -
```

Now, you can press RETURN to leave the flags this way or enter different flag values.

Search

Syntax

```
Srange list
```

Purpose

Search the *range* specified for the *list* of bytes specified and print the start address of the string.

Comments

The *list* may contain one or more bytes, each separated by a space or comma.

If the *list* contains more than one byte, only the first address of each byte string found is returned.

If the *list* contains only one byte, all addresses of the byte in the range are displayed.

When you enter

```
SCS:100 110 41
```

DEBUG might return the response

```
04BA:0104
04BA:010D
-
```

Trace

Syntax

```
T[=address][ value]
```

Purpose

Execute one instruction and display the contents of all registers, flags, and the decoded instruction.

Comments

If the optional *=address* is entered, tracing occurs at the *=address* specified.

If *=address* includes the segment designation, then both CS and the instruction pointer are specified.

If *=address* omits the segment designation, only the instruction pointer is specified.

The optional *value* causes DEBUG to execute and trace the number of steps specified by *value*. The T function uses the hardware trace mode of the 8086 or 8088 microprocessor. Consequently, the user may also trace instructions stored in ROM.

If you enter

```
T
```

and press the **RETURN** key, DEBUG returns a display of the registers, flags, and decoded instruction for that one instruction. Assume that the current position is 04BA:011A; then DEBUG might return the display:

```
AX=0000  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0BCB  SI=0B0B  DI=0000
DS=0CCA  ES=0CCA  SS=0CCA  CS=0CCA  IP=0102   NV UP EI PL NZ NA PE NC
0CCA:0102 0000           ADD      [BX+SI],AL                      DS:0B0B=5A
```

Now enter

```
T=011A 10
```

and press **RETURN**. DEBUG executes sixteen instructions beginning at 011A in the current segment and then displays all registers and flags for each instruction as it is executed. The display scrolls away until the last instruction is executed. Then the display stops, and you can see the register and flag values for the last few instructions performed.

Remember that CTRL-NUM LCK suspends the display at any point, so that you can study the registers and flags for any instruction.

Unassemble

Syntax

```
U[address][ L value]
U[range]
```

Purpose

Disassemble bytes and display the source statements that correspond to them, along with addresses and byte values.

Comments

The display of disassembled code looks like a listing for an assembled file.

If you enter the U function without parameters, 20 hexadecimal bytes are disassembled to show corresponding instructions.

If you enter the U function with the *range* parameter, then DEBUG disassembles all bytes in the range.

If there are fewer bytes in the *range* than the number displayed when U is entered without parameters, then only the bytes in the *range* are displayed.

If you enter the U function with the *address* parameter, then DEBUG disassembles the default number of bytes, beginning at the *address* specified.

If you enter the U function with the *address* L *value* parameters, then DEBUG disassembles all bytes beginning at the address specified for the number of bytes specified by *value*. Entering the U function with the *address* L *value* parameters overrides the default limit (20H bytes).

When you enter

```
U04BA:100 L10
```

and press **RETURN**, DEBUG disassembles 16 bytes beginning at address 04BA:0100:

```
04BA:0100  206472    AND  [SI+72],AH
04BA:0103  69        DB   69
04BA:0104  7665      JBE  016B
04BA:0106  207370    AND  [BP+DI+70],DH
04BA:0109  65        DB   65
04BA:010A  63        DB   63
04BA:010B  69        DB   69
04BA:010C  66        DB   66
04BA:010D  69        DB   69
04BA:010E  63        DB   63
04BA:010F  61        DB   61
```

If you enter

```
U04BA:0100 0108
```

and press **RETURN**, DEBUG disassembles only those bytes between address 04BA:0100 and address 04BA:0108. The display shows

```
04BA:0100  206472    AND  [SI+72],AH
04BA:0103  69        DB   69
04BA:0104  7665      JBE  016B
04BA:0106  207370    AND  [BP+DI+70],DH
```

Unassemble

If you enter

```
U04BA:100 120
```

and press **RETURN**, DEBUG disassembles and displays all the bytes between address 100 and address 120.

If, however, you enter the function

```
UCS:100 L 20
```

and press the **RETURN** key, all of the bytes beginning at address CS:100, for the specified twenty bytes, are disassembled and displayed. Notice that you can specify the alphabetic segment register in place of the four digit segment address without effecting the results.

If the bytes in some addresses are altered, the disassembler alters the instruction statements. The U function can be entered for the changed locations, the new instructions viewed, and the disassembled code used to edit the source file.

Write

Syntax

```
W[address| drive record record]]
```

Purpose

Write the file being debugged to a disk file.

Comments

If only the W appears, BX:CX must already be set to the number of bytes to be written; the file is written beginning from CS:100.

If the W function is given with just an address, then the file is written beginning at that address.

If a G or T function was used, BX:CX must be reset before using the Write function without parameters.

NOTE: If a file is loaded and modified, the name, length, and starting address are all set correctly to save the modified file as long as the length has not changed.

The file must have been named either with the DEBUG invocation command or with the N function (see the description of the Name function). Both the invocation and the N function format a file name properly in the normal format of a file control block at CS:5C.

DEBUG

Write

If the W function is given without parameters, BX:CX must be set to the number of bytes to be written. Then, DEBUG writes the BX:CX number of bytes to the disk file. The debugged file is written to the disk from which it was loaded. This means that the debugged file is written over the original file that was loaded into memory.

If the W function is given with parameters, the write begins from the memory address specified; and the file is written to the *drive* specified, (where drive names are represented by numbers— 0=A:, 1=B:, 2=C:, 3=D:, 4=E:, 5=F:, 6=G:, and 7=H:). DEBUG writes the file beginning at the logical record number specified by the first *record*; and the write continues until the number of sectors specified in the second *record* have been written.

CAUTION: Writing to absolute sectors is *EXTREMELY* dangerous because the process bypasses the file handler.

When you enter

```
W
```

and press the **RETURN** key, DEBUG writes out the file to disk, then displays the DEBUG prompt:

```
W
-
```

In the example,

```
WCS:100 1 37 2B
```

DEBUG writes out the contents of memory, beginning with the address CS:100 to the disk in drive B:. The data written out starts in disk logical record number 37H and consists of 2BH records. When the write is complete, DEBUG displays the prompt:

```
WCS:100 1 37 2B
-
```

Error Messages

During the DEBUG session, you may receive any of the following error messages. Each error terminates the DEBUG function with which it is associated but does not terminate the DEBUG command itself or return control to MS-DOS.

ERROR CODE	EXPLANATION
BF	Bad Flag The user attempted to alter a flag, but the characters entered were not one of the acceptable pairs of flag values. See the Register function for the list of acceptable flag entries.
BP	Too many Breakpoints More than ten breakpoints have been specified as parameters to the G function. Reenter the Go function with ten or fewer breakpoints.
BR	Bad Register The R function was entered with an invalid register name. See the Register function for the list of valid register names.
DF	Double Flag Two values have been entered for one flag. You may specify a flag value only once per RF function.

Part V

Winchester Command Guide

Winchester Command Guide

This guide provides a comprehensive description of the four commands in your MS-DOS software package that cause a change to a Winchester disk.

Knowledge of these commands is not necessary for users who do not have a Winchester disk or for Winchester disk users who do not wish to change or move their Winchester disk.

These commands are explained in terms that can be understood by users who have had no prior microcomputer experience. The only prerequisite for using this guide and the commands it describes is that you understand the concepts and perform the necessary procedures in Part I, "Preparation Guide" of this manual. Familiarity with some sections of Part II, "Primary Feature Guide" is also helpful for using these commands.

The sequence of the chapters in this guide corresponds to sequence in which these commands will probably be used. However, there are circumstances in which one of these commands could be used alone or in which this sequence could be changed.

The Winchester Command Guide consists of the following four chapters:

Chapter 16, "PREP"—This chapter includes explanations of PREP's purpose, entry form, preliminary concepts, usage examples, and error messages. The PREP command initializes a Winchester disk so it is ready for use as a mass storage device. You will rarely (if ever) have to use PREP.

Chapter 17, "PART"—This chapter includes explanations of PART's purpose, entry forms, preliminary concepts, usage examples, and error messages. The PART command is run whenever you intend to change the arrangement of MS-DOS partitions on the Winchester disk. Additionally, PART permits you to select a default boot partition.

Chapter 18, "SHIP"—This chapter includes explanations of SHIP's purpose, entry forms, preliminary concepts, usage examples, and error messages. The SHIP command is run whenever you intend to physically move the Winchester disk. SHIP moves the read/write heads of a Winchester disk to a position where they are less likely to destroy media or stored data if jolted during physical transit.

Chapter 19, "DETECT"—This chapter includes explanations of DETECT's purpose, entry form, preliminary concepts, usage examples, and error messages. The DETECT command examines a Winchester disk and isolates unusable sectors so that they will not be accessed by MS-DOS.

NOTE: These commands conform to the definition of commands found in Chapter 5, "Command Features" of Part II, "Primary Feature Guide," and they can be entered according to the command entry rules explained in Chapter 5.

If you need information about commands that help with original software program development, refer to Part IV, "Program Development Command Guide."

If you need information about commands that perform functions that do not require a Winchester disk, refer to Part III, "Primary Command Guide."

Chapter 16

PREP

Purpose

To magnetically prepare a Winchester disk surface for mass storage of software and/or data.

Entry Form

```
PREP
```

Preliminary Concepts

The PREP utility initializes a Winchester disk so that it is ready for use as a mass storage device by the microcomputer. PREP is seldom (if ever) used by most users.

CAUTION: Using PREP will destroy all software and/or data stored on your Winchester disk. Do not use PREP until you have transferred your Winchester disk files to floppy disks. Run PREP only if there are an unreasonable number of disk access errors that you cannot correct with the DETECT utility.

After you use the PREP utility, you will have to reset and reboot the system with a bootable floppy disk.

Winchester Disk Features

Winchester disks come in a variety of sizes and configurations, but they all have common features. The central feature and core of a Winchester disk device is a rigid platter. The typical platter consists of a nonmagnetic metal (generally aluminum) disk, coated with a thin plating of ferric oxide or cobalt. This platter itself is the Winchester disk, as opposed to the floppy disk, which has a plastic core with a thin coating of a similar magnetic substrate.

Platters

Winchester disk platters are generally sealed to prevent contamination from the environment (such as dust, smoke, dirt, or hair) from contaminating a platter's surface or read/write head. Winchester disks are available in a variety of sizes and with one or more platters. Winchester disks can be either fixed or removable. The *fixed* Winchester disk is permanently mounted inside the device, but *removable* Winchester disks come in disk packs or cartridges and may be removed or interchanged.

Read/Write Heads

The read/write heads are electromagnets that slide back and forth above the surface of Winchester disk platters. Read/write heads are extremely close to (but not touching) platter surfaces. The movement of a Winchester disk drive's read/write heads between the hub of the platters and the edge of the platters is called *stepping*. Therefore, this movement is measured by an amount known as the *step rate*.

Logical Winchester Disk Division

Winchester disks can be divided into several logical subunits called disk *partitions*. This is partly because large quantities of storage locations are easier to deal with if they are subdivided. The various subdivisions help speed storage and retrieval of data.

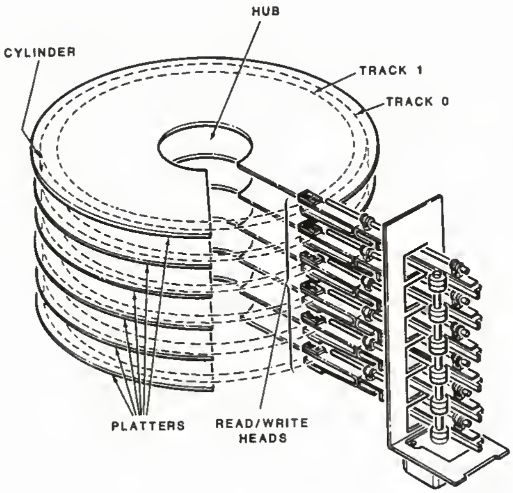


Figure 16.1. A Typical Winchester Disk

Sector

A *sector* is the basic unit of data organization for disk drive devices. Like floppy disks, Winchester disks are divided into sectors. Winchester disk sectors under MS-DOS are 512 bytes long, just as they are with floppy disks.

Tracks and Cylinders

Each recording surface of a Winchester disk platter is also divided into concentric rings called *tracks*, which are similar to the tracks of a floppy disk (see Figure 16.1). The Winchester disks initialized by PREP are formatted with 17 sectors per track. A *cylinder* is a collection of all tracks that are located the same distance from the outer edge of each recording surface. Winchester disk read/write heads can access all of the data stored on a particular cylinder without any stepping movement.

For example, if a Winchester disk drive has four read/write heads, the drive can access a cylinder of 68 sectors (4 tracks times 17 sectors) without stepping or use of different sets of heads. This amounts to a total of 34,816 bytes (34 kilobytes) read or written.

Prompts

The PREP utility helps you to do the following:

- initialize the surface of the Winchester disk,
- test the data retention capabilities of the Winchester disk media, and
- isolate disks sectors that may be questionable.

When invoked, PREP displays a message in the following form:

```
        PREP version 1.75

The PREP utility helps you to:

  * Initialize surface of Winchester disk
  * Test data retention capabilities of Winchester disk media
  * Isolate questionable disk sectors
```

PREP will prompt you for the Winchester drive unit number. Then PREP displays messages as it operates on the disk.

```
Do you wish to proceed with PREP (Y/N)?
```

Typing an N at this prompt ends the PREP utility and returns you to the system prompt.

Typing a Y causes PREP to display the following prompt:

```
Please type P to proceed
```

Typing any response other than P ends the PREP utility and returns you to the system prompt.

Typing a P causes PREP to display the following prompt:

```
Winchester drive unit number (0-7):
```

Entering the Winchester drive unit number causes PREP to continue operation.

PREP Operations

PREP prepares the disk by performing three operations in sequence: initializing the disk, testing media, and initializing the Reserved Winchester Area.

Initializing the Disk

PREP begins to prepare the surface of your selected Winchester disk after you enter the drive unit number to continue operation.

PREP Operations

After you have responded to the `Winchester drive unit number (0-7):` prompt, PREP initializes the surface of the disk for the test that will follow. During initialization you will see the message:

```
Initializing the disk...
```

This initialization is similiar to some of the activities of the FORMAT command in that it magnetically records a map of all sectors on the disk surface. When the surface has been initialized, the message shows:

```
Initializing the disk...completed
```

and the seven test passes begin.

Testing Media

PREP performs seven test passes to check the integrity of the disk's storage capability. During each pass, PREP writes a predetermined code to each sector (the drive light will flicker) and then reads back that code to verify that it remained correct (the drive light will appear as constantly on). PREP keeps you informed of its progress by displaying the message:

```
Media test in progress, pass n
```

Where *n* is the number (in the range 1-7) of the pass that it is currently conducting.

Be patient. This PREP operation requires approximately 45 minutes per 5 megabytes of Winchester disk capacity.

PREP uses a different code on each pass it makes through the test. If PREP finds unusable sectors, it stores the address of these sectors and later places these sector addresses into a bad sector table.

Initializing the Reserved Winchester Area

After completing the media tests, PREP records and verifies the Reserved Winchester Area (see The Reserved Winchester Area section of this chapter) on the first several sectors of the Winchester disk. During this operation, PREP displays the following message:

```
Initializing the disk...
```

When PREP adds the word completed to the end of this display and displays the system prompt, then all PREP operations are complete. The display should appear as follows:

```
Initializing the disk...completed

A>
```

NOTE: You will not be able to access any partition after using PREP until you reset the system and boot up with a bootable floppy disk.

The Reserved Winchester Area

NOTE: Information concerning the Reserved Winchester Area is not essential for use of the PREP utility or the Winchester disk. This information is provided for users who wish to obtain a deeper understanding of the operations that PREP performs in order to prepare a Winchester disk.

When the PREP utility is run, it records units of Winchester support software on the first sector and the second to last cylinder of the Winchester disk. These software units are collectively known as the *Reserved Winchester Area*. They are recorded on the Winchester disk during PREP's reserved area initialization operation.

The Reserved Winchester Area

The most important Winchester support software units are the boot record, the partition table, and the bad sector tables. The boot record and partition tables are located on the first sector of the Winchester disk (head 0, cylinder 0, sector 1). The bad sector tables are located on the next to last cylinder of the Winchester disk.

These units are vital to you during Winchester bootup because they help you to access a particular partition after you access the Winchester disk itself. Users of MS-DOS with the Winchester disk also use these data structures to make unusable media (bad sectors) inaccessible before the FORMAT command is run.

The Boot Record

The *boot record* is Winchester support software that helps locate the partition to be booted after entry of a Winchester disk bootup command.

PREP records the boot record on the first sector of the Winchester disk during initialization of the Reserved Winchester Area.

When you enter a Winchester disk bootup command, the microcomputer will load the boot record. The boot record differs from the boot loader, which is another software unit that participates in the booting of a Winchester partition. The boot record is recorded at the beginning of the Winchester disk, whereas boot loaders are recorded at the beginning of individual partitions. (For more information on the boot loader, refer to Chapter 6, "Bootup Features." For more information on the component loading sequence, refer to Chapter 9, "System Component Features.")

Boot Indicator Byte

Once within RAM, the boot record begins to access a partition. The partition that is accessed is determined by the boot indicator byte. The *boot indicator* byte is used by the boot record to determine if one of the partitions contains a loadable operating system.

The boot record loads the first sector from the partition and checks for a valid signature in the loader. If the loader is valid, the control is transferred to the loader.

When the boot record finds a partition that matches the specified boot indicator byte, the boot record loads the first sector of that partition into RAM. If the accessed partition contains MS-DOS, then the MS-DOS boot loader program will begin to execute the remainder of the bootup operation.

The Partition Table

The *Partition Table* is a Winchester support unit that contains information about each partition on the disk. It contains the following items for each of the 4 defined partitions:

- starting cylinder, head & sector,
- ending cylinder, head & sector,
- operating system ID,
- boot indicator,
- number of sectors preceding the partition, and
- number of sectors occupied by the partition.

The structure of the Partition Table is illustrated in Table 16.1.

The column headed by 1BE and ending with 1FA (in Table 17.2) represents the hexadecimal offsets to the boot record.

The Reserved Winchester Area

Table 16.1. The Partition Table Structure

1BE	Partition #1 Start	Boot	H	S	CYL
1C2	Partition #1 End	Sys	H	S	CYL
1C6	Partition #1 rel. sec.	Low word rel. sec.		High word rel. sec.	
1CA	Partition #1 num. secs.	Low word num. secs.		High word num. secs.	
1CE	Partition #2 Start	Boot	H	S	CYL
1D2	Partition #2 End	Sys	H	S	CYL
1D6	Partition #2 rel. sec.	Low word rel. sec.		High word rel. sec.	
1DA	Partition #2 num. secs.	Low word num. secs.		High word num. secs.	
1DE	Partition #3 Start	Boot	H	S	CYL
1E2	Partition #3 End	Sys	H	S	CYL
1E6	Partition #3 rel. sec.	Low word rel. sec.		High word rel. sec.	
1EA	Partition #3 num. secs.	Low word num. secs.		High word num. secs.	
1EE	Partition #4 Start	Boot	H	S	CYL
1F2	Partition #4 End	Sys	H	S	CYL
1F6	Partition #4 rel. sec.	Low word rel. sec.		High word rel. sec.	
1FA	Partition #4 num. secs.	Low word num. secs.		High word num. secs.	

The Partition Table begins at the end of the boot record. Each entry in the table is 16 bytes long and contains the starting and ending cylinder (CYL), sector (S), and head (H) for each of the 4 possible partitions as well as the number of sectors preceding the partition and the number of sectors occupied by the partition. For a more detailed explanation, refer to the Partition Table section of Chapter 17, "PART."

The boot indicator byte is used by the boot record to locate the partition that was specified in the bootup command. The PART utility places the sum of 80 hexadecimal plus the Winchester disk unit number in the corresponding partitions boot indicator byte, thereby marking the user selected partition bootable (all other partition's boot indicators are set to zero at the same time).

The presence of a value from 80 to 87 hexadecimal (instead of 00) in a partition's boot indicator tells the microcomputer to load the sector whose location is contained in the following 3 bytes. That particular sector will be the actual boot loader for the selected operating system and will be responsible for the rest of the system's loading process.

The Bad Sector Table

The *bad sector table* is an ordered list of sectors on the disk that contain unusable media. The information in the bad sector table enables MS-DOS to avoid bad sectors (unusable media) when it accesses a partition during everyday activities. The bad sector table is located on the next to last cylinder on the Winchester disk.

The bad sector table can indicate as many as 168 bad sectors. Each bad sector is recorded in a three-byte entry consisting of the head, cylinder, and sector. Entries that do not contain bad sectors are filled with three zeroes.

The structure of the bad sector table is explained in Table 16.2.

Bad Sector Table Entries

During media testing, PREP maintains a record of the location of all the bad sectors (sectors containing unusable media) that it finds on the disk. Then, during initialization of the Reserved Winchester Area, PREP creates a table of all bad sectors and records two copies of this table on the disk.

Table 16.2. Bad Sector Table Entries

BYTES	BAD SECTOR TABLE ENTRY
3	Head, cylinder, and sector of 1st bad sector found by PREP
3	Head, cylinder, and sector of 2nd bad sector found by PREP
3	Head, cylinder, and sector of 3rd bad sector found by PREP
3	Head, cylinder, and sector of 4th bad sector found by PREP
.	.
.	.
.	.
3	Head, cylinder, and sector of 168th bad sector found by PREP
3	Last entry in table (always contains 000)
2	Checksum
2	Reserved bytes
1	Interleave byte (disk formatted)
512	Total for each bad sector table

Bad Sector Table Verification

The primary copy of the bad sector table, called Bad Sector Table A, is used unless some of its contents have been damaged since it was recorded. The backup copy of the bad sector table, called Bad Sector Table B, is used if Bad Sector Table A is damaged. Each copy is recorded several sectors apart, to decrease the chance that both could be damaged simultaneously.

A checking code, called a *checksum*, is calculated by PREP for each of the copies of the bad sector table before PREP records these bad sector table copies on the Winchester disk. The results of these checksums are recorded in the bad sector table itself.

Then, when execution of PREP is repeated on the same Winchester disk, verification checksums are performed to verify that the bad sector tables have not changed since the original checksums were performed.

If Bad Sector Table A cannot be read, or if the results of the second checksum of Bad Sector Table A differ from the results of the original checksum, then Bad Sector Table B is read.

If Bad Sector Table B cannot be read, or if the results of the second checksum of Bad Sector Table B differ from the results of the original checksum, then no bad sector table information will be available in the Reserved Winchester Area. If the FORMAT command is then used on a partition of this disk, it will assume that the disk has no bad sectors and format without avoiding any bad sectors. If the DETECT command is used on this disk, it will find no bad sector table to which it can append new bad sectors. Therefore, it will search the disk for all bad sectors and create a new bad sector table.

Error Messages

```
Bad sector count exceeded for this drive.
```

EXPLANATION: The upper bound limit for bad sectors has been exceeded. This could indicate a hardware malfunction. Run PREP again. If this error message reappears after repeated use of PREP, then contact your Technical Consultation or the dealer from whom you purchased your microcomputer for assistance.

PREP

Error Messages

```
Error -- Drive capacity > 32 megabytes!
```

EXPLANATION: PREP has calculated that the Winchester disk drive connected to the Winchester controller is larger than the maximum allowable size of 32 megabytes.

```
Error during formatting of the drive.
```

EXPLANATION: This message could indicate a hardware malfunction on the Winchester controller board. Contact your Technical Consultation or the dealer from whom you purchased your microcomputer for assistance.

```
Boot sector is bad.
```

EXPLANATION: A bad sector error has occurred in the boot sector of the Winchester disk which contains the boot record. This could indicate a hardware malfunction. Run PREP again. If this error message reappears after repeated use of PREP, contact your Technical Consultation for assistance.

```
Unable to communicate with the Winchester controller
```

EXPLANATION: PREP cannot locate the Winchester controller. This could mean that the Winchester controller is not firmly plugged into the bus, that the drive cable connectors are not securely fastened, or that the controller has a hardware malfunction. Check to see that the controller card and all cable connectors are secure, and run PREP again.

Chapter 17
PART

Purpose

The PART utility is run whenever you intend to change the arrangement of partitions on the Winchester disk. Additionally, PART permits you to select a default boot partition.

Entry Form

```
PART
```

Preliminary Concepts

The PART utility enables you to perform the following functions:

- select a default boot partition,
- allocate space on the disk for a specific partition by specifying the start and end cylinder, and
- delete a partition allocated to MS-DOS.

You do not necessarily need to run PART in order to use your Winchester disk because an MS-DOS partition was prepared on the disk by PREP before it was shipped.

The PART utility is recorded on one of the DOS distribution disks which are shipped with the microcomputer package. The PREP, SHIP, and DETECT utilities are also recorded on one of these DOS disks.

CAUTION: Any changes you make to the partition table using PART can destroy all existing data on your Winchester disk. Therefore, you should use the BACKUP utility to back up all necessary data from all affected partitions before you use PART.

Preliminary Concepts

Winchester disks have large storage capacities. To make practical use of all this storage space, it can be divided into partitions. You can establish up to four partitions on your Winchester disk and record a different operating system on each partition.

A partition behaves like a floppy disk in most operations, because you can access a partition's data and/or software by entering commands that refer to the drive name to which that particular partition has been assigned.

NOTE: The exact capacity of your Winchester disk drive is determined by the drive's manufacturer and by the amount of usable disk space remaining after unusable space has been marked as unusable by software, such as the PREP utility.

PART Operation

The PART utility enables you to change the status of partition features by typing different kinds of entries in response to prompts. During the course of using PART, the screen displays a *Partitioning Menu* which shows the types of each partition, the start and end cylinder of each partition and the amount of disk space allocated to each partition (in percentages and kilobytes). As you make changes to disk partitions, features of the partitioning menu will change to reflect the changes you make to the partitions (see the section entitled The Partitioning Menu within this chapter).

Before we continue with the major features of PART, some information is needed on how to invoke this utility.

Invoking PART

PART is recorded on one of the disks supplied with your DOS package as an executable file.

PART Operation

To invoke the PART utility, enter the following command at the system prompt:

```
PART
```

then press **RETURN**. When PART is first invoked, an identification message and warning appear as follows:

```
                 PART version 2.00

      The PART utility helps you to change the arrangement of the
DOS  partitions on  your  Winchester disk.  PART displays a table
showing the types of each partition,  the start cylinder, the end
cylinder and the amount of disk space allocated to each partition
(in percentages and in kilobytes).

CAUTION:  Using PART can destroy all DOS files on your Winchester
disk.

Do  not  use  PART  until you  have transferred  copies  of  your
Winchester disk files to floppy disks.

Do you wish to continue (Y/N)?_
```

If you enter an **N** at the `Do you wish to continue (Y/N)?` prompt, then PART exits to the DOS. If you enter a Y, then PART displays the following screen (referred to as the *Drive Select menu*) if you have only one Winchester drive attached:

```
  F1 - Partition Winchester drive 1

  E  - Exit to DOS

Enter selection <F1 or E>
```

NOTE: Although most systems will have only one Winchester drive, PART supports the partitioning of up to eight Winchester drives. PART alters its Drive Select menu depending on how many drives are attached to your system. If you have eight Winchester drives, the menu would appear as follows:

```
  F1 - Partition Winchester drive 1
  F2 - Partition Winchester drive 2
  F3 - Partition Winchester drive 3
  F4 - Partition Winchester drive 4
  F5 - Partition Winchester drive 5
  F6 - Partition Winchester drive 6
  F7 - Partition Winchester drive 7
  F8 - Partition Winchester drive 8

  E  - Exit to DOS

Enter selection <F1, F2, F3, F4, F5, F6, F7, F8 or E>
```

If you enter **E**, PART will exit to the DOS. If you press one of the partitioning selections (**F1** or **F1-F8**), a Partitioning menu similar to one of the following two is displayed:

```
     Partition      Start      End        Size in     Percentage
       Type        Cylinder   Cylinder   Kilobytes   of the Disk
     ---------     --------   --------   ---------   -----------
1.   DOS              0         304        10369        100.0%
2.   Unallocated
3.   Unallocated
4.   Unallocated

Winchester unit = 0
Default boot partition = 1
Maximum cylinder Number = 303
Minimum DOS allocation = 32 Kilobytes


  F1 - Select default boot partition
  F2 - Allocate DOS partition
  F3 - Delete DOS partition
  E  - Exit

Enter selection <F1, F2, F3 or E>_
```

Figure 17.1. Partitioning Menu #1

```
       Partition      Start      End        Size in      Percentage
         Type        Cylinder   Cylinder    Kilobytes    of the Disk
       ---------     --------   --------    ---------    -----------
1.     non-DOS          0         304         10369         100.0%
2.     Unallocated
3.     Unallocated
4.     Unallocated

Winchester unit = 0
Default boot partition = 1
Maximum cylinder Number = 303
Minimum DOS allocation = 32 Kilobytes


  F1 - Select default boot partition
  F2 - Allocate DOS partition
  E  - Exit

Enter selection <F1, F2, or E>_
```

Figure 17.2. Partitioning Menu #2

The Partitioning menu presented in Figure 17.1 is the screen that is displayed when you have the option of allocating a DOS partition. This would be the case if there is at least one unallocated partition or if a DOS partition exists. That is to say, you can allocate an unallocated partition, as well as change an already allocated DOS partition. This screen is also displayed when you have the option of deleting at least one existing DOS partition (non-DOS and unallocated partitions cannot be deleted).

The Partitioning menu presented in Figure 17.2 is the screen that is displayed when you have no DOS partitions to delete (no DOS partitions exist).

The Partitioning Menu

The Partitioning menu allows you to select and run three partitioning options. These options are:

PART Operation

- Selecting a default boot partition.
- Allocating space to a DOS partition.
- Deleting a DOS partition.

The Partitioning menu also displays the types associated with each partition, the start and end cylinders of each partition, and the amount of disk space allocated to each partition (in kilobytes and percentages).

As you make changes to disk partitions, features of the Partitioning menu will change to reflect the changes you make to the partitions.

When displayed, the basic form of the Partitioning menu will appear as follows:

```
     Partition     Start     End        Size in     Percentage
       Type       Cylinder  Cylinder    Kilobytes   of the Disk
     ---------    --------  --------    ---------   -----------
1.   DOS             0        304         10369        100.0%
2.   Unallocated
3.   Unallocated
4.   Unallocated

Winchester unit = 0
Default boot partition = 1
Maximum cylinder Number = 303
Minimum DOS allocation = 32 Kilobytes


  F1 - Select default boot partition
  F2 - Allocate DOS partition
  F3 - Delete DOS partition
  E  - Exit

Enter selection <F1, F2, F3 or E>_
```

The *Partition Type* heading refers to one of three “types” of partitions. These three types are as follows:

- DOS—This is an MS-DOS Winchester disk partition.

- Non-DOS—This is a partition allocated to some other operating system. PART will not modify space allocated to this partition in any way.

- Unallocated—An entry with this type of partition is free to be used for creating a DOS partition.

The *Start Cylinder* heading is the first Winchester disk cylinder that is allocated to the associated partition.

The *End Cylinder* heading is the last Winchester disk cylinder that is allocated to the associated partition.

The *Size in Kilobytes* heading refers to the size in kilobytes of the partition.

The *Percentage of the Disk* heading indicates what percent of the disk the particular partition occupies.

The *Winchester unit* field indicates which Winchester drive the user has selected to partition. Note that the unit number is one less than the drive number.

The *Default boot partition* field indicates which partition (1-4) of the Winchester disk the user has selected as the default boot partition.

The *Maximum cylinder Number* field refers to the total number of cylinders on the selected Winchester disk.

The *Minimum DOS allocation* field indicates the minimum size in kilobytes that the user can select to make up a partition.

PART Operation

The F1-F3 selections at the bottom of the screen are function key selections which represent partitioning options available to the user. These options are as follows:

- F1 allows you to select a default boot partition.
- F2 allows you to allocate a DOS partition.
- F3 allows you to delete a DOS partition.

Pressing the E selection (located at the very bottom of the partitioning options) returns you to the Drive Select menu.

When one of these function keys are pressed, the screen will display prompts which are appropriate to the partitioning option you selected.

Selecting a Default Boot Partition

The default boot partition is usually the partition that you intend to use most often for booting up. Any partition that is selected as the default boot partition is the partition that will be automatically accessed when you enter a Boot command without specifying a partition (see Chapter 6, "Bootup Features," for details on the Boot command).

To select a default bootup partition, do the following:

- Invoke PART (see the section Invoking PART in this chapter).

- Select to partition a Winchester drive (unit 0). For our example, your microcomputer will have only one Winchester drive.

PART Operation

After you have selected to partition a Winchester drive, a screen similar to the following will be displayed:

```
     Partition    Start      End        Size in     Percentage
       Type      Cylinder  Cylinder    Kilobytes    of the Disk
     ---------   --------  --------    ---------    -----------
1.   Non-DOS        0        100          3433          33.1%
2.   Unallocated
3.   Unallocated
4.   Unallocated

Winchester unit = 0
Default boot partition = 4
Maximum cylinder Number = 303
Minimum DOS allocation = 32 Kilobytes

  F1 - Select default boot partition
  F2 - Allocate DOS partition
  F3 - Delete DOS partition
  E  - Exit

Enter selection <F1, F2, F3 or E>_
```

By looking at the *Default boot partition* field, you can see that partition 4 is the current default boot partition. If you wanted to select another partition (1, for example) as the default boot partition, press **F1** at the `Enter selection <F1, F2, F3 or E>_` prompt. Pressing E returns you the Drive Select menu.

After you press F1, the following prompt will appear at the current cursor position:

```
Select partition (1 - 4) or RETURN for none:_
```

PART Operation

For our example, enter **1** at this prompt (you may also choose partitions 2, 3 or 4) to select the default boot partition. After you select 1 as the default boot partition, a screen similar to the following will display:

```
     Partition     Start      End       Size in     Percentage
       Type       Cylinder  Cylinder   Kilobytes    of the Disk
     ---------    --------  --------   ---------    -----------
1.   Non-DOS         0        100        3433          33.1%
2.   Unallocated
3.   Unallocated
4.   Unallocated

Winchester unit = 0
Default boot partition = 1
Maximum cylinder Number = 303
Minimum DOS allocation = 32 Kilobytes


  F1 - Select default boot partition
  F2 - Allocate DOS partition
  F3 - Delete DOS partition
  E  - Exit

Enter selection <F1, F2, F3 or E>_
```

Notice that the *Default boot partition* field has changed to 1, reflecting which partition is the current default boot partition.

NOTE: All three types of partitions (DOS, non-DOS, and unallocated) may be selected as the default boot partition.

Allocating a Partition

By specifying the start and end cylinder for each partition, PART enables you to determine the size of each partition. As the start and end cylinders are selected, PART will automatically calculate the size of the partition in kilobytes, as well as the percentage of space the partition occupies on the disk.

NOTE: Only unallocated and DOS partitions may be allocated. Non-DOS partitions may not be allocated.

In order to allocate a partition, do the following:

- Invoke PART (see the section Invoking PART in this chapter).
- Select to partition a Winchester drive (for our example, your microcomputer will have only one Winchester drive).

After you have selected to partition a Winchester drive, a screen similar to the following will be displayed:

```
      Partition     Start      End       Size in      Percentage
        Type       Cylinder  Cylinder   Kilobytes     of the Disk
      ---------    --------  --------   ---------     -----------
1.    Unallocated
2.    Unallocated
3.    Unallocated
4.    Unallocated

Winchester unit = 0
Default boot partition = None
Maximum cylinder Number = 303
Minimum DOS allocation = 32 Kilobytes


  F1 - Select default boot partition
  F2 - Allocate DOS partition
  E  - Exit

Enter selection <F1, F2 or E>_
```

To allocate a partition, press **F2** at the `Enter selection <F1, F2 or E>_` prompt.

After you press F2, the following prompt appears:

```
Select Partition (1-4):_
```

PART Operation

For this example, you will create a DOS partition on the first partition of the disk that is 50 cylinders in size. Therefore, enter 1 at this prompt (you may also choose any partition of type DOS or Unallocated) to select the partition to allocate.

After you select the partition at the `Select Partition (1-4):_` prompt, the following prompt appears at the current cursor position:

```
Start Cylinder:_
```

Here, you would enter the starting cylinder number for the first partition on the Winchester disk. For our example, you will create a DOS partition which is 50 cylinders in size. Therefore, enter a **0** at this prompt and then press **RETURN**.

NOTE: If an entry for the start cylinder is invalid because it is non-numeric or too large (greater than the maximum cylinder number allowed), then a bell is sounded and you will be re-prompted for the information.

After you select the starting cylinder number and press RETURN, PART will display the following prompt:

```
End Cylinder:_
```

Enter the ending cylinder number for the first partition on the Winchester disk. Since, for this example, you are creating a DOS partition which is 50 cylinders in size, enter **50**.

Before you press RETURN to enter the end cylinder number, take a look at your screen. It will look similar to the following display:

```
     Partition     Start     End      Size in     Percentage
       Type       Cylinder  Cylinder  Kilobytes   of the Disk
     ---------    --------  --------  ---------   -----------
1.   Unallocated
2.   Unallocated
3.   Unallocated
4.   Unallocated

Winchester unit = 0
Default boot partition = None
Maximum cylinder Number = 303
Minimum DOS allocation = 32 Kilobytes

Select Partition (1-4): 1
Start Cylinder: 0
End Cylinder: 50_
```

After entering the end cylinder number, press **RETURN**. Now, take another look at your screen. It will look similar to the following display:

```
     Partition     Start     End      Size in     Percentage
       Type       Cylinder  Cylinder  Kilobytes   of the Disk
     ---------    --------  --------  ---------   -----------
1.   DOS             0         50       1733         16.7%
2.   Unallocated
3.   Unallocated
4.   Unallocated

Winchester unit = 0
Default boot partition = None
Maximum cylinder Number = 303
Minimum DOS allocation = 32 Kilobytes


  F1 - Select default boot partition
  F2 - Allocate DOS partition
  F3 - Delete DOS partition
  E  - Exit

Enter selection <F1, F2, F3, or E>_
```

PART Operation

Notice that PART automatically calculated the size of the partition in kilobytes (1733). It also calculated the percentage of space the partition now occupies on the disk (16.7%).

Error Conditions

When allocating a partition on a Winchester disk, you may encounter some error conditions. If your entries are valid (within the range of 0 to the maximum cylinder number and does not overlap any non-DOS or DOS partitions), the screen is redrawn and updated to reflect your changes. However, if you have entered invalid cylinder numbers, one of four error messages (see Table 17.1) will display on the screen.

NOTE: The error messages presented in Table 17.1 illustrate two principle points:

- Disk space cannot be allocated to more than 1 partition.
- DOS partitions can only be allocated from 32 kilobytes to 32 megabytes in size.

Table 17.1. Allocation Error Messages

```
1.    Error - Requested allocation overlaps DOS partition.

      Press any key to continue. . .

2.    Error - Requested allocation overlaps non-DOS partition.

      Press any key to continue. . .

3.    Error - Requested allocation is less than required DOS
      minimum.

      Press any key to continue. . .

4.    Error - Requested allocation exceeds DOS maximum of 32
      megabytes.

      Press any key to continue. . .
```

Error message number 1 (Table 17.1) occurs when the cylinders of one partition intersect or "overlap" the cylinders of another DOS partition. This situation is illustrated in the following display:

```
     Partition      Start     End       Size in      Percentage
       Type        Cylinder  Cylinder   Kilobytes    of the Disk
     ---------     --------  --------   ---------    -----------
1.   DOS              0        50         1733          16.7%
2.   Unallocated
3.   Unallocated
4.   Unallocated

Winchester unit = 0
Default boot partition = None
Maximum cylinder Number = 303
Minimum DOS allocation = 32 Kilobytes

Select Partition (1-4): 2
Start Cylinder: 25
End Cylinder: 65_

Error - Requested allocation overlaps DOS partition

Press any key to continue. . .
```

Notice in this display that the area between the starting and ending cylinders of the second partition (25 and 65) overlap the area between the starting and ending cylinders of the first partition (0 and 50). This condition cannot be tolerated by PART.

Error message number 2 (Table 17.1) will appear under basically the same conditions as error message 1—the one exception is that the area between the starting and ending cylinders of one partition overlap the area between the starting and ending cylinders of a non-DOS partition.

Error message number 3 (Table 17.1) will appear when an allocated partition does not meet the DOS minimum size of 32 kilobytes.

Error message number 4 (Table 17.1) will appear when an allocated partition exceeds the DOS maximum size of 32 megabytes.

NOTE: With these PART error situations, the error messages are displayed and the operation of PART is suspended until you press any key. This allows you to examine the input and correct it for the next time through.

Deleting a Partition

In addition to allowing you to select a default boot partition and allocate a partition, the PART utility also permits you to delete a partition.

NOTE: The PART utility will only allow you to delete a DOS partition. Unallocated and non-DOS partitions cannot be deleted. If you try to delete an invalid partition, a bell is sounded and PART waits for another input.

To delete a DOS partition, do the following:

- Invoke PART (see the section Invoking PART in this chapter).
- Select to partition a Winchester drive (for our example, your microcomputer will have only one Winchester drive).

PART Operation

After you have selected to partition a Winchester drive, a screen similar to the following will be displayed:

```
     Partition      Start      End        Size in      Percentage
       Type        Cylinder   Cylinder   Kilobytes    of the Disk
     ---------     --------   --------   ---------    -----------
1.   DOS              0          50        1733           16.7%
2.   Unallocated
3.   Unallocated
4.   Unallocated

Winchester unit = 0
Default boot partition = None
Maximum cylinder Number = 303
Minimum DOS allocation = 32 Kilobytes


  F1 - Select default boot partition
  F2 - Allocate DOS partition
  F3 - Delete DOS partition
  E  - Exit

Enter selection <F1, F2, F3 or E>_
```

Since only DOS partitions may be deleted, the only partition you are free to delete in this screen display is the first.

To delete a partition, press **F3** at the `Enter selection <F1, F2, F3 or E>_` prompt.

NOTE: If there are no DOS partitions present on the Winchester disk, then the `F3 - Delete DOS partition` option will not display on the screen.

After you press F3, the following prompt will appear at the current cursor position:

```
Select Partition (1-4):_
```

Enter the number of the partition that is to be deleted at this prompt. In this case, enter **1** to delete the first partition.

PART Operation

After you select the partition to be deleted, a screen similar to the following will display:

```
     Partition     Start     End        Size in     Percentage
       Type       Cylinder  Cylinder   Kilobytes    of the Disk
     ---------    --------  --------   ---------    -----------
1.   Unallocated
2.   Unallocated
3.   Unallocated
4.   Unallocated

Winchester unit = 0
Default boot partition = None
Maximum cylinder Number = 303
Minimum DOS allocation = 32 Kilobytes


  F1 - Select default boot partition
  F2 - Allocate DOS partition
  E  - Exit

Enter selection <F1, F2 or E>_
```

Notice that when a DOS partition is deleted, it automatically becomes an unallocated partition. Also, take note that when no DOS partitions are present on the Winchester disk, the `F3 - Delete DOS partition` option is no longer displayed.

The Exit Function

After you have completed the partitioning operation (selecting a default boot partition, allocating a partition and/or deleting a partition), select the E (Exit function) at the `Enter Selection <F1, F2, F3 or E>` prompt. When you do, a screen similar to the following will display (if you have only one Winchester drive attached):

```
  F1 - Partition Winchester drive 1

  M  - Make changes, exit to MFM-150
  A  - Abort all changes, exit to DOS

Enter selection <F1, M or A>_
```

PART Operation

NOTE: Although most systems will have only one Winchester drive, PART supports partitioning of up to eight Winchester drives. PART alters its menu, depending on how many drives are attached to your system. If you have eight drives, the menu will look as follows:

```
   F1 - Partition Winchester drive 1
   F2 - Partition Winchester drive 2
   F3 - Partition Winchester drive 3
   F4 - Partition Winchester drive 4
   F5 - Partition Winchester drive 5
   F6 - Partition Winchester drive 6
   F7 - Partition Winchester drive 7
   F8 - Partition Winchester drive 8

   M  - Make changes, exit to MFM-150
   A  - Abort all changes, exit to DOS

Enter selection <F1,F2,F3,F4,F5,F6,F7,F8 M or A>_
```

If at this time you want to return to the Partitioning menu to make additional changes to the Winchester partitions, then you should make one of the function key selections.

If you have made changes during partitioning and wish to exit the partitioning session, press **M**.

NOTE: After you press M, you must reset and reboot the system. When changes are made, MS-DOS must be reloaded into memory after using PART. Resetting ensures the integrity of data on the Winchester disk.

If you wish to abort all changes or have made no changes at all, press **A** and you will be returned to the DOS.

The Partition Table

The *Partition Table* is a Winchester support unit that contains the following information for each of the four defined partitions:

- starting cylinder, head & sector,
- ending cylinder, head & sector,
- operating system ID,
- boot indicator,
- number of sectors preceding the partition, and
- number of sectors occupied by the partition.

The structure of the Partition Table is illustrated in Table 17.2.

Table 17.2. The Partition Table Structure

1BE	Partition #1 Start	Boot	H	S	CYL
1C2	Partition #1 End	Sys	H	S	CYL
1C6	Partition #1 rel. sec.	Low word rel. sec.		High word rel. sec.	
1CA	Partition #1 num. secs.	Low word num. secs.		High word num. secs.	
1CE	Partition #2 Start	Boot	H	S	CYL
1D2	Partition #2 End	Sys	H	S	CYL
1D6	Partition #2 rel. sec.	Low word rel. sec.		High word rel. sec.	
1DA	Partition #2 num. secs.	Low word num. secs.		High word num. secs.	
1DE	Partition #3 Start	Boot	H	S	CYL
1E2	Partition #3 End	Sys	H	S	CYL
1E6	Partition #3 rel. sec.	Low word rel. sec.		High word rel. sec.	
1EA	Partition #3 num. secs.	Low word num. secs.		High word num. secs.	

Table 17.2 (cont'd.) The Partition Table Structure

1FE	Partition #4 Start	Boot	H	S	CYL
1F2	Partition #4 End	Sys	H	S	CYL
1F6	Partition #4 rel. sec.	Low word rel. sec.		High word rel. sec.	
1FA	Partition #4 num. secs.	Low word num. secs.		High word num. secs.	

The Partition Table is stored at the end of the boot record on sector 1, cylinder 0, head 0 of the Winchester disk. The *boot record* is Winchester support software that helps locate the partition to be booted after entry of a Winchester disk Boot command.

When you enter a Winchester disk Boot command, the microcomputer will load the boot record. Once within RAM, the boot record begins to access a partition. This partition is determined by the *boot indicator* byte.

The *boot indicator* byte (*Boot* in Table 17.2) is used by the boot record to determine if one of the partitions contains a loadable operating system. This byte will contain the drive number the disk partition resides on if the disk partition is bootable. This byte is modified at boot time and used as a flag for which partition to boot, as opposed to flagging which partition is bootable.

The *system indicator* byte (*Sys* in Table 17.2) is used to identify which operating system a partition belongs to. PART is only able to identify partitions with system indicators 0 (for unallocated partitions) and 1 (for DOS partitions). Any other system indicator byte indicates a non-DOS partition.

The column headed by *1BE* and ending with *1FA* (in Table 17.2) represents the hexadecimal offsets to the boot record.

The *Partition Start* field (*Start* in Table 17.2) indicates the values associated with the head (H), sector (S), and cylinder (CYL) of the partition's starting point on the Winchester disk.

The Partition Table

The *Partition End* field (*End* in Table 17.2) indicates the values associated with the head, sector, and cylinder (*H*, *S*, and *CYL* respectively in Table 17.2) of the partitions ending point on the Winchester disk.

The *Relocation Sector* field (*rel. sec.* in Table 17.2) represents a double-word count of the number of sectors that precede the partition on the Winchester disk (a word represents 2 bytes or 16 bits. A double-word count represents 4 bytes or 32 bits).

The *Number of Sectors* field (*num. secs.* in Table 17.2) represents a double-word count of the number of sectors that belong to the corresponding partition.

At the offset, 1FE on the Partition Table in the first sector is a byte value (signature) that, if present, indicates a valid Partition Table exists. This byte value will be a 55 hexadecimal at offset 1FE and AAH (hexadecimal) at offset 1FF.

One other important feature of PART has to do with the signature at the end of the boot record. When PART finds a valid boot record (that is, if a signature is correct), it takes the Partition Table it finds and uses it for further operations. If the signature is not correct, then PART will assume that the table is invalid and assign a new Partition Table of all *unallocated* partitions. PART will then write out a new boot record when the user exits the program with changes.

Error Messages

```
Cannot read the boot sector
```

EXPLANATION: This error message will appear when you are selecting a Winchester drive to partition and PART cannot read the boot sector. This error is usually caused by a bad disk. Press any key to continue. When this error message appears, contact your Technical Consultation for assistance.

`Cannot write the boot sector`

EXPLANATION: This error message will appear when you are exiting a partitioning session with changes and select the M option. PART cannot write the changes you have made to the appropriate boot sector. This error is usually caused by a bad disk. Press any key to continue. When this error message appears, contact your Technical Consultation for assistance.

```
Disk is fully allocated to non-DOS partitions, no partitioning is
possible.
```

EXPLANATION: After selecting a drive to partition, the Partitioning menu appears and this message is displayed. PART waits for a key (any) to be pressed. After a key has been pressed, PART returns to the Drive Select menu. This message is displayed when the disk cannot be partitioned by PART. PART only operates on partitions of type DOS and unallocated. If the disk is fully allocated to non-DOS partitions, then partitions that are unallocated cannot be allocated any space.

`Unable to communicate with the Winchester controller`

EXPLANATION: This error message is displayed immediately after you make a partition drive selection from one of PART's main menus. After the message is displayed, PART immediately exits to DOS. This message means that PART cannot locate the Winchester controller. This could mean that the Winchester controller is not firmly plugged into the microcomputer bus, all of the drive cable connectors are not securely fastened, or the controller has a hardware malfunction. Check to see that the controller card and all cable connectors are secure, and run PART again. If this message appears again, then contact your Technical Consultation for assistance.

Chapter 18

SHIP

Purpose

SHIP moves the read/write heads of a Winchester disk to a position where they cannot destroy stored data in the event of excessive physical shock experienced during physical transit. Run SHIP before physically moving your Winchester disk.

Entry Form

```
SHIP
```

Preliminary Concepts

Winchester disks are sensitive precision instruments that can be easily affected by physical shock or impact. The data stored on a Winchester disk is also vulnerable. Because of this vulnerability, you should take special precautions when shipping your Winchester disk, or even when moving the disk across the room.

The SHIP command enables you to protect your Winchester disk and the data on the disk. SHIP affords this protection by moving the disk's read/write heads towards the hub of the Winchester disk platters. When in this position, the heads and platters are less likely to be damaged by platter vibration that can be caused by physical shock. This position of the read/write heads is known as the *shipping cylinder* and will be referred to as such throughout this text.

Although platters can be caused to vibrate at their outside edges, the platter area near the hub is rigid enough to inhibit vibration. Therefore, the heads and platters are safer when the heads are near the hub.

SHIP

Preliminary Concepts

Run SHIP whenever you intend to physically move the unit containing your Winchester disk.

NOTE: The Winchester controller card causes the read/write heads to move to cylinder zero the first time you access the Winchester disk after power up. Therefore, the head positioning caused by SHIP will remain in effect only until you turn the disk on again and access it.

Prompts

By simply responding yes to one prompt, SHIP will position the read/write heads on every Winchester drive on your system to the shipping cylinder. You should use SHIP before moving your Winchester drive.

Winchester Drive Unit

When invoked, SHIP displays a message in the following form:

```
             SHIP version 1.00

The SHIP utility helps you to:

   * Position the read/write heads of the Winchester disk
     at a safe location for subsequent transportation
     of the disk unit.

After running SHIP, no further operation on your computer will be
possible.  At that time,  turn off the power to the computer  and
prepare it for shipping.

Do you wish to continue (Y/N):
```

Shipping Cylinder

If you repond yes to the `Do you wish to continue` prompt, SHIP will move the read/write heads to the shipping cylinder. Then the system will display the following message:

```
Heads moved to shipping cylinder. Turn off your microcomputer and
prepare it for shipping.
```

The heads will remain at the shipping cylinder while you turn off the microcomputer and move it.

Error Message

```
Unable to communicate with the Winchester controller
```

EXPLANATION: SHIP cannot locate the Winchester controller. This could mean that the Winchester controller is not firmly plugged into the microcomputer bus, that the drive cable connectors are not securely fastened, or that the controller has a hardware malfunction. Check to see that the controller card and all cable connectors are secure, and run SHIP again.

Chapter 19

DETECT

Purpose

Examines a Winchester disk and isolates unusable sectors so that they will not be accessed by MS-DOS.

Entry Form

```
DETECT
```

Preliminary Concepts

The DETECT utility examines your Winchester disk for any bad sectors (media imperfections) that have occurred since the disk was shipped or since the PREP utility was last used. Then, DETECT adds the addresses of these bad sectors to a list of bad sectors that was recorded on the Winchester disk when PREP was run. This list is called the *bad sector table*.

The DETECT utility is shipped on the Winchester Utility Disk.

The PREP utility has already been run on all Winchester disks. (Refer to Chapter 16, "PREP," for information on the bad sector table.)

NOTE: The bad sectors found by DETECT will not be locked out by MS-DOS until the partition has been formatted using the FORMAT utility.

Bad Sectors

Bad sectors are media imperfections that can cause hard errors during Winchester disk access operations. *Hard errors* are conditions in which an operation failed after a number of repeated attempts. When you encounter a hard error, you must abort in order to return to the operation being performed. Unfortunately, corruptions may have infiltrated into your file as the hard error developed. Also, much of your work to this point may be unretrievable.

The DETECT utility enables you to prevent hard errors from occurring in the future if these errors were caused by bad sectors. If DETECT finds new bad sectors (between 1 and 168), it adds them to the bad sector table that was originally created by the PREP utility. If DETECT finds more than 168 bad sectors, an error message will be displayed.

Then, the next time you format a partition, FORMAT will take into consideration the newly-acknowledged bad sectors. FORMAT will set up sector boundaries that will prevent usage of the bad sectors during all operations that occur after the formatting operation.

However, you might also obtain hard errors during Winchester disk access due to the following other problems:

- excessive physical shock exerted on the disk (more than 5 G's for a period of time greater than 11 milliseconds)
- entry of foreign material (such as smoke) into the Winchester disk chamber
- malfunction of the Winchester controller card
- temporary loss of power to the disk

If one of these problems causes a hard error, then the disk might not have any more bad sectors for DETECT to find. In such a case, you should back up the files from your Winchester disk and use the PREP utility. If you still encounter hard errors after using PREP, contact your Technical Consultation for assistance.

DETECT does not destroy any of the data on the Winchester disk. However, if DETECT does isolate bad sectors, we recommend that after DETECT, you use BACKUP to copy all files from the partition on which the bad sectors occurred. Then, you should use the FORMAT utility on the partition on which the bad sectors occurred. Finally, you should use RESTORE to replace the backed up files on this partition.

Prompts

DETECT will prompt you for the Winchester drive that you wish to use. It will then ask for known bad sectors. You can determine a bad sector's number by observing the sector address in the system error message. If you know that a certain sector is bad, type in its logical sector number at this prompt. Typing in a zero and then pressing RETURN causes DETECT to begin its detection process. DETECT will read the entire disk looking for bad sectors. Any bad sectors will be added to the bad sector table.

When invoked, DETECT displays a message in the following form:

```
         DETECT version x.xx

The DETECT utility helps you to:

   * Locate sectors that have failed since you last ran PREP

Do you wish to proceed with DETECT (Y/N)?
```

DETECT

Preliminary Concepts

At the `Do you wish to proceed with DETECT (Y/N)?` prompt, press Y to continue with the utility or press any other key to exit to the system.

If you press Y to continue, DETECT displays the following prompt:

```
Winchester drive unit number (0-7):
```

This prompt is asking you to enter the number associated with the Winchester drive unit you wish to use. After you enter that number, DETECT displays the following prompt:

```
Enter bad sector address, or zero to end:
```

When bad sectors are encountered during disk access operations, MS-DOS displays a hard error message that is slightly different from the floppy disk hard error message. This message appears in the following form:

```
<type> error <I/0action> drive <d>
Sector address of error is <nnnn>
Abort,Retry,Ignore:
```

where `<type>` indicates the type of problem that caused the error condition. This problem could be worded as:

```
Write Protect
SEEK
DATA
SECTOR NOT FOUND
WRITE FAULT
```
or
```
DISK;
```

`<I/0 action>` identifies the operation that was being performed when the error occurred. This operation could be worded as:

```
reading
or
writing
```

<d> is the name of the drive to which the partition was assigned when an error was encountered on the partition; and

<nnnn> is the logical hexadecimal address of the sector on which the hard error occurred. (Logical sector addresses begin with the first sector on the entire Winchester disk, which is sector 0000.)

We recommend that you record the sector address of the error when this hexadecimal value is displayed. Also record the number of the partition and the drive unit number of the Winchester disk on which the error(s) occurred. This information can become vital when the `Invalid HEX value, Try again:` error message appears.

At the `Enter bad sector address, or zero to end:` prompt, enter the address of the logical sector at which the error(s) occurred, or press the digit zero (0) to begin media verification.

If you enter the address of a logical sector, then DETECT will continue to display the prompt:

```
Enter bad sector address, or zero to end:
```

until you enter a zero.

DETECT Operation

When DETECT begins to search for bad sectors (after you have typed a zero at the `Enter bad sector address, or zero to end:` prompt), DETECT displays the following message:

```
Beginning detection...
```

When DETECT is finished detecting bad sectors, it will display the following message:

```
Beginning detection...Completed
```

If DETECT found no bad sectors during the operation, it will also display the following message:

```
No bad sectors detected.
```

If DETECT found a number (1-168) of bad sectors during its search, it will display the following message:

```
Bad sectors located. Tables modified.
```

NOTE: The words `Tables modified` will not appear in this message if DETECT is unsuccessful in recording the new bad sector information at the end of the bad sector table.

If DETECT finds more than 168 bad sectors on the Winchester disk, it will display the following message:

```
Bad sector count exceeded for this drive.
```

DETECT Followup Activities

After you use the DETECT utility, the data stored on your Winchester disk will still be intact (except the data that was recorded over bad sectors). However, the addition to the bad sector table that DETECT provides will not be put to use until you use the FORMAT utility on the newly-verified media.

The bad sectors that DETECT found will not become inaccessible until FORMAT is used on the partition that contained the bad sectors. FORMAT will redefine the sector boundaries of the partition so that the bad sectors cannot be accessed.

If you have not already done so, boot up to a floppy disk or partition other than the partition just verified.

We recommend that you use the BACKUP utility to copy all of the files from the partition to floppy disks as soon as possible after using DETECT.

Then, with all of the files safely stored on floppy disk media, use the FORMAT utility. Specify the drive that has been assigned the partition on which the bad sector(s) occurred.

After formatting this partition, use the RESTORE utility to copy the backed up files back to the Winchester disk partition.

You should take the earliest possible opportunity after using DETECT to perform these activities to ensure the safety of your stored data.

Error Messages

```
Bad sector count exceeded for this drive.
```

EXPLANATION: The upper bound limit of 168 bad sectors has been exceeded. This could indicate a hardware malfunction. Run DETECT again. If this error message appears after repeating DETECT, then run PREP. If this error message appears after running PREP, then contact your Technical Consultation for assistance.

DETECT

Error Messages

```
Error -- Drive capacity > 32 megabytes!
```

EXPLANATION: DETECT has calculated that the Winchester drive connected to the Winchester controller card is larger than the maximum allowable size of 32 megabytes.

```
Error -- Unable to read boot code from partition
```

EXPLANATION: The boot code on the specified partition is either not present or it has developed a bad sector. Boot up from another drive. Then run DETECT, BACKUP, FORMAT, and RESTORE in this sequence on the partition where the error occurred. If that partition is totally unavailable, you may need to run the PREP utility.

```
Invalid HEX value, Try again:
```

EXPLANATION: Value entered was not a valid hexadecimal number or the value entered was outside of the possible range. By referring to your log of hard error messages, double check the appropriate hex value and reenter. If you have not maintained a log of hard error messages, you must now try to recreate the error so as to obtain the actual hex number.

```
Unable to communicate with the Winchester controller
```

EXPLANATION: DETECT cannot locate the Winchester controller. This could mean that the Winchester controller is not firmly plugged into the microcomputer bus, all of the drive cable connectors are not securely fastened, or that the controller has a hardware malfunction. Check to see that the controller card and all cable connectors are secure, and run DETECT again. If this error message appears after repeating DETECT, then run PREP. If this error message appears after running PREP, then contact your Technical Consultation for assistance.

Index

Index

& (ampersand) as a LIB command character, 13.7, 13.8, 13.10, 13.14, 13.17
* (asterisk)
  as the EDLIN prompt, 12.3, 12.4
  to identify current line in EDLIN, 12.5
  as a LIB command character, 13.1, 13.7, 13.8, 13.9 – 13.10, 13.14, 13.16
  as a wildcard character, 1.15
@ (at sign) with the NEW TEMPLATE function, 12.40
\ (backslash), 7.6, 7.16
  in the QUIT INPUT or VOID function, 12.37
  root directory, 5.22
  to separate directories, 7.6, 7.16
, (comma) as a delimiter, 5.7, 5.11
. (dot) as directory name, 7.5
.. (dot dot) as directory name, 7.5
= (equal sign) as a delimiter, 5.7, 5.11
> (greater than symbol)
  as the default system prompt, 5.6
  as the system prompt, 5.3, 5.4
. (period) as an interline command parameter with EDLIN, 12.5
| (pipe) with MORE, 11.239, 11.240
+ (plus sign)
  as a delimiter, 5.7, 5.11
  as a LIB command character, 13.1, 13.7, 13.8, 13.9 – 13.10, 13.14, 13.15
  with LINK, 14.15 – 14.16
# (pound sign) as an EDLIN interline command parameter, 12.6
? (question mark)
  with ASSIGN, 11.18
  with EDLIN, 12.6
  as a wildcard character, 1.15
; (semicolon)
  as a delimiter, 5.7, 5.11
  as a LIB command character, 13.7, 13.8, 13.10, 13.16 – 13.17
  with LINK, 14.16
/A (abort print) switch with PRINT, 11.248, 11.250
/A (after date) switch
  with BACKUP, 11.30 – 11.31
  with RESTORE, 11.281, 11.287 – 11.288
/B (before date) switch
  with BACKUP, 11.31 – 11.32
  with RESTORE, 11.281, 11.288 – 11.289
/B (binary comparison) switch with FC, 11.164, 11.169, 11.174
/C (clear directory and file allocation tables) switch with FORMAT, 11.186, 11.190

Index

/C (count lines) switch with FIND, 11.176, 11.178, 11.179
/C (ignore case of letters) switch with FC, 11.164, 11.169 – 11.170
/C*n*(copies) switch with PRINT, 11.248, 11.250 – 11.251
/C (not to search directories) switch with SEARCH, 11.307
/D (directory master) switch
    with BACKUP, 11.32 – 11.33
    with RESTORE, 11.281, 11.289 – 11.290
/D (list directories) switch with SEARCH, 11.307, 11.308
/DSALLOCATE switch with LINK, 14.22
/E (exception files) switch
    with BACKUP, 11.33
    with RESTORE, 11.281, 11.290
/F (files) switch with TREE, 11.330 – 11.336
/F (fix) switch with CHKDSK, 11.59, 11.60, 11.62, 11.63
/F (flat restoration) switch with RESTORE, 11.281, 11.290 – 11.291
/F (format silent) switch with BACKUP, 11.33 – 11.34
/F (form feed) switch with PRINT, 11.248, 11.251
/G (global subdirectories) switch with BACKUP, 11.34 – 11.35
/HIGH switch with LINK, 14.23
/I (ignore letter case) switch with FIND, 11.176, 11.177, 11.179
/LINENUMBERS switch with LINK, 14.23
/L (list directory) switch
    with BACKUP, 11.35 – 11.36
    with RESTORE, 11.281, 11.291 – 11.292
/L*n*(left margin) switch with PRINT, 11.248, 11.251
/MAP switch with LINK, 14.23
/M (initialize disk as single-sided) switch with FORMAT, 11.187, 11.190
/M (map output drive) switch with RESTORE, 11.281, 11.292 – 11.293
/N (no formatting) switch with BACKUP, 11.36
/N (number lines) switch with FIND, 11.176, 11.178, 11.179 – 11.180
/N (suppress onscreen prompt) switch with FORMAT, 11.187, 11.190
/O (overwrite files) switch with RESTORE, 11.281, 11.293
/PAUSE switch with LINK, 14.24
/P*n*(page length) switch with PRINT, 11.248, 11.251
/P (page mode) switch with DIR, 11.137, 11.139, 11.143, 11.144
/Q (query each) switch
    with BACKUP, 11.37
    with RESTORE, 11.281, 11.294
/R*n*(right margin) switch with PRINT, 11.248, 11.252
/R (reverse sort) switch with SORT, 11.318, 11.319
/R (review selected files) switch
    with BACKUP, 11.37 – 11.38
    with RESTORE, 11.281, 11.294 – 11.295
/S (copy system files) switch with FORMAT, 11.187, 11.190
/S (spool print on) switch with PRINT, 11.248, 11.252
/STACK switch with LINK, 14.24
/T (display a graphic representation of the directory structure) switch with SEARCH, 11.307, 11.308
/T (terminate) switch with PRINT, 11.248, 11.252

Index

/T (today's date) switch
  with BACKUP, 11.38
  with RESTORE, 11.281, 11.295
/V (variant lines) switch with FIND, 11.176, 11.180
/V (verbose) switch with CHKDSK, 11.59, 11.60, 11.63
/V (verify files) switch
  with BACKUP, 11.38 – 11.39
  with RESTORE, 11.281, 11.295 – 11.296
/V (verify formatting) switch with FORMAT, 11.187, 11.190 – 11.191
/V (verify function on) switch with DISKCOPY, 11.150 – 11.154
/V (verify) switch with COPY, 11.114, 11.117
/W (compress "whites") switch with FC, 11.164, 11.170
/W (wide display mode) switch with DIR, 11.137, 11.139, 11.143, 11.144
/W (written files only) switch with BACKUP, 11.39
/Z (reading/copying Zenith Data Systems' CP/M formatted disks) switch with RDCPM, 11.264, 11.266
/8 (8 sectors per track) switch with FORMAT, 11.187, 11.190
/# (number of lines) switch with FC, 11.164, 11.170
/ + *n*(start sort at column *n*) switch with SORT, 11.318, 11.320

A
Absolute path name, 7.6
Alignment defined, 14.5
Ambiguous commands defined, 6.8 – 6.9
Ampersand (&) as a LIB command character, 13.7, 13.8, 13.10, 13.14, 13.17
ANSI.SYS as a terminal driver, 11.2
  cursor control with, 11.4 – 11.6
  erasing with, 11.7
  installation of, 11.3
  mode of operation with, 11.8 – 11.9
  purpose of, 11.2
  reassignment of keyboard key functions with, 11.10
  use of, 11.3
APPEND LINES as an EDLIN interline command, 12.3, 12.5, 12.6 – 12.7
Application programs defined, 1.2
APPLY as a command, 10.2, 10.15
  command line entry of, 11.12 – 11.15
  entry forms of, 11.11 – 11.12
  error messages of, 11.16
  preliminary concepts of, 11.12
  purpose of, 11.11
Argument as a command line parameter, 5.6, 5.7, 5.48
Assemble with DEBUG, 15.5, 15.9 – 15.11
ASSIGN as a command, 10.2, 10.11
  command line entry of, 11.18 – 11.20
  entry form of, 11.17

Index

Index

ASSIGN as a command (continued)
  error messages of, 11.21 – 11.22
  preliminary concepts of, 11.17 – 11.18
  purpose of, 11.16
Asterisk (*)
  as the EDLIN prompt, 12.3, 12.4
  to identify current line in EDLIN, 12.5
  as a LIB command character, 13.1, 13.7, 13.8, 13.9 – 13.10, 13.14, 13.16
  as a wildcard character, 1.15
At sign (@) with the NEW TEMPLATE function, 12.40
AUTOEXEC.BAT file, 5.22
  creating an, 5.23 – 5.24
Automatic command entry to create batch files, 5.21 – 5.22, 5.48

B
Background printing. *See* Spool printing with PRINT
Backslash (\)
  with QUIT INPUT or VOID function, 12.37
  root directory, 5.22
  to separate directories, 7.6, 7.16
BACKUP as a command, 10.2, 10.9, 10.12, 10.13, 10.15
  advanced concepts of, 11.43 – 11.44
  backup volumes with, 11.44 – 11.45
  command line entry of, 11.28
  destination file with, 11.29
  entry forms of, 11.23 – 11.24
  error messages of, 11.48 – 11.52
  extension assignment with, 11.44 – 11.45
  help screen of, 11.27 – 1.28
  interactive entry prompt and response with, 11.28
  internal directory of, 11.46 – 11.48
  preliminary concepts of, 11.24 – 11.27
  purpose of, 11.23
  running, from a batch file, 11.42
  source file specification with, 11.29
  specifying a series of source files with, 11.39 – 11.40
  switches with, 11.30 – 11.39
  using EDLIN with, 11.43
  using exception files with, 11.40 – 11.41
  using the interactive method with, 11.41 – 11.42
Backup file defined, 11.24
Backup procedure
  for general disks, 3.4 – 3.8
  for MS-DOS distribution disks, 3.1 – 3.3
  purpose of, 3.1
Bad sectors
  isolation of, with DETECT, 19.1 – 19.7
  with RECOVER, 11.212 – 11.218
Bad sector table
  with DETECT, 19.1, 19.2
  with PREP, 16.11 – 16.13

- Batch commands, entry of, 5.25
- Batch files
  - creation of, 5.24
  - examples of, 5.32 – 5.36
  - executing, 5.30 – 5.32
  - with PAUSE, 11.245 – 11.247
  - with REM, 11.274 – 11.276
  - with replaceable parameters, 5.29 – 5.30
- Batch processing
  - to specify replaceable parameters, 5.2B – 5.29
  - tutorial, 5.26 – 5.28
- Batch-processing resident commands
  - ECHO as, 5.25, 10.3, 10.7, 11.156 – 11.159
  - FOR as, 5.25, 10.4, 10.7, 11.183 – 11.186
  - GOTO as, 5.25, 10.4, 10.7, 11.199 – 11.201
  - IF as, 5.26, 10.4, 10.7, 11.201 – 11.206
  - PAUSE as, 5.26, 10.4, 10.7, 11.245 – 11.247
  - REM as, 5.26, 10.5, 10.7, 10.B, 11.274 – 11.276
  - SET as, 5.3, 10.5, 10.8, 10.1B, 11.309 – 11.314
  - SHIFT as, 5.26, 10.5, 10.7, 11.314 – 11.317
- Binary files
  - altering of, with DEBUG, 15.2
  - with FC, 11.163- 11.175
- Boot command, 6.1, 6.19
  - automatic, 6.5 – 6.6
  - entry values of, 6.4
  - error messages of, 6.15 – 6.1B
  - manual, 6.1 – 6.2, 6.6 – 6.9, 6.12 – 6.13
  - methods of, 6.5
  - results of, 6.14 – 6.15
  - sequence of, 6.3 – 6.4
- Boot indicator byte defined, 16.9
- Boot loader defined, 9.4
- Boot record defined, 16.B
- BREAK as a command, 10.2, 10.17
  - advanced concepts of, 11.54
  - command line entry of, 11.53 – 11.54
  - entry forms of, 11.53
  - preliminary concepts of, 11.53
  - purpose of, 11.52
- Buffer defined, 5.14
- Bytes defined, 1.5

C

- CHDIR or CD as a command, 7.5, 7.7, 7.12, 10.2, 10.10, 10.11, 10.14
  - changing a directory with, 7.12
  - command line entry of, 11.55 – 11.56
  - displaying the status of the current working directory with, 11.57 – 11.5B
  - entry forms of, 11.54 – 11.55
  - error message of, 11.5B

Index

CHDIR or CD as a command (continued)
  preliminary concepts of, 11.55
  purpose of, 11.54
  specifying a new directory by name with, 11.56 – 11.57
  specifying a new directory by shorthand notation with, 11.57
CHKDSK as a command, 10.2, 10.10, 10.11, 10.17
  advanced concepts of, 11.62 – 11.63
  command line entry of, 11.60 – 11.62
  entry form of, 11.59
  error messages of, 11.63 – 11.67
  preliminary concepts of, 11.59
  purpose of, 11.58
  switches with, 11.59, 11.60, 11.62, 11.63
CIPHER as a command, 10.2, 10.9, 10.16
  command line entry of, 11.68 – 11.69
  copying an encrypted file to another file with, 11.72 – 11.73
  creating an encrypted file with, 11.70 – 11.72
  decrypting a file with, 11.73 – 11.74
  displaying an encrypted file with, 11.72
  encrypting an existing file with, 11.70
  entry forms of, 11.67
  error message of, 11.74
  preliminary concepts of, 11.68
  purpose of, 11.67
Class defined, 14.4
CLS as a command, 10.2, 10.8
  command line entry of, 11.75
  entry form of, 11.74
  preliminary concepts of, 11.74
  purpose of, 11.74
Combine type defined, 14.5
Comma (,) as a delimiter, 5.7, 5.11
COMMAND as a command, 10.2, 10.8
  advanced concepts of, 11.79 – 11.80
  command line entry of, 11.78
  entry form of, 11.75 – 11.76
  error messages of, 11.80 – 11.82
  preliminary concepts of, 11.76 – 11.77
  purpose of, 11.75
COMMAND.COM properties as physical component, 9.7
  initialization portion of, 9.8
  resident portion of, 9.7
  transient portion of, 9.8 – 9.9
  to use resident commands, 5.2, 5.13
Command defined, 5.1
Command interruption, 5.21, 5.38, 5.49
Command line
  defined, 1.20, 5.6
  editing a, 5.13 – 5.21
  parameters of, 5.6 – 5.10
  requirements for entry of, 1.21 – 1.22, 5.11 – 5.13

Index

Command line buffer, 5.14
Command line method
  to invoke LIB, 13.7, 13.8 – 13.11
  to invoke LINK, 14.13, 14.17
Command line template, 5.14 – 5.15, 5.48
Command processor, interruption of, 5.38
Command prompt method
  to invoke LIB, 13.7 – 13.8
  to invoke LINK, 14.13, 14.14 – 14.15
Command prompts
  with LIB, 13.7 – 13.8, 13.13 – 13.15
  with LINK, 14.19 – 14.22
Commands, 1.18
  APPLY (transient), 10.2, 10.15, 11.11 – 11.16
  ASSIGN (transient), 10.2, 10.11, 11.16 – 11.22
  BACKUP (transient), 10.2, 10.10, 10.12, 10.13, 10.15, 11.23 – 11.52
  boot, 6.1 – 6.19
  BREAK (resident), 5.3, 10.2, 10.17, 11.52 – 11.54
  CHDIR or CD (resident), 5.3, 7.5, 7.7, 7.12, 10.2, 10.10, 10.11, 10.14, 11.54 – 11.58
  CHKDSK (transient), 5.4, 10.2, 10.10, 10.11, 10.17, 11.58 – 11.67
  CIPHER (transient), 10.2, 10.9, 10.16, 11.67 – 11.74
  CLS (resident), 5.3, 10.2, 10.8, 11.74 – 11.75
  COMMAND (transient), 5.4, 10.2, 10.8, 11.75 – 11.82
  COMP (transient), 10.2, 10.10, 10.17, 11.83 – 11.94
  CONFIGUR (transient), 5.4, 10.2, 10.18, 11.95 – 11.113
  convenience, 10.8 – 10.9
  COPY (resident), 5.3, 7.8, 7.10, 10.3, 10.12, 10.13, 10.16, 11.114 – 11.126
  CTTY (resident), 5.3, 10.3, 10.18, 11.126 – 11.128
  d:, 5.3
  data filtering, 10.9
  data/software analysis, 10.10
  data/software change, 10.11
  data/software creation, 10.10 – 10.12
  data/software movement, 10.12 – 10.14
  DATE (resident), 5.3, 10.3, 10.8, 10.10, 11.128 – 11.132
  DEBUG (transient), 5.4, 15.1 – 15.41
  DEL or ERASE (resident), 5.3, 7.5, 7.10, 10.3, 10.11, 10.16, 11.132 – 11.136
  DETECT (transient), 5.4, 19.1 – 19.8
  directory, 7.7 – 7.8, 10.14 – 10.15
  DIR (resident), 5.3, 5.5, 7.2, 7.10, 10.3, 10.10, 11.137 – 11.145
  DISKCOMP (transient), 5.4, 10.3, 10.10, 10.17, 11.145 – 11.149
  DISKCOPY (transient), 3.3, 3.5 – 3.8, 5.4, 10.3, 10.12, 10.13, 10.16, 11.150 – 11.155
  ECHO (resident, batch-processing), 5.3, 5.25, 10.3, 10.7, 11.156 – 11.159
  EDLIN (transient), 5.4, 12.1 – 12.46
  EXE2BIN (transient), 11.159 – 11.162

Index

Commands (continued)
  EXIT (resident), 5.3, 10.3, 10.8, 11.162 – 11.163
  FC (transient), 5.5, 10.3, 10.10, 10.17, 11.163 – 11.175
  FIND (transient), 5.5, 10.3, 10.9, 11.176 – 11.182
  FOR (resident, batch-processing), 5.3, 5.25, 10.4, 10.7, 11.183 – 11.186
  FORMAT (transient), 5.5, 10.4, 10.11, 10.12, 10.14, 10.18, 11.186 – 11.199
  GOTO (resident, batch-processing), 5.3, 5.25, 10.4, 10.7, 11.199 – 11.201
  hard copy, 10.13
  IF (resident, batch-processing), 5.3, 5.26, 10.4, 10.7, 11.201 – 11.206
  LIB (transient), 5.4, 13.1 – 13.21
  LINK (transient), 5.4, 14.1 – 14.28
  for manipulating directories, 7.7 – 7.8, 7.20
  manipulation, 10.15 – 10.16
  MAP (transient), 10.4, 10.8, 10.11, 11.207 – 11.211
  MKDIR or MD (resident), 5.3, 7.7, 7.13, 10.4, 10.12, 10.14, 11.218 – 11.221
  MODE (transient), 10.4, 10.18, 11.221 – 11.238
  MORE (transient), 5.4, 10.4, 10.9, 11.239 – 11.241
  PART (transient), 5.4, 17.1 – 17.22
  PATH (resident), 5.3, 7.7, 7.8 – 7.9, 10.4, 10.12, 10.14, 11.241 – 11.244
  paths and resident, 7.10 – 7.11
  paths and transient, 7.8 – 7.9
  PAUSE (resident, batch-processing), 5.3, 5.26, 10.4, 10.7, 11.245 – 11.247
  PREP (transient), 5.4, 16.1 – 16.14
  PRINT (transient), 5.4, 10.4, 10.15, 11.247 – 11.258
  PROMPT (resident), 5.3, 10.4, 10.8, 11.258 – 11.261
  PSC (transient), 10.4, 10.15, 11.261 – 11.263
  RDCPM (transient), 5.4, 10.4, 10.13, 10.16, 11.263 – 11.268
  RECOVER (transient), 5.4, 10.4, 10.14, 10.16, 10.17, 11.268 – 11.274
  REM (resident, batch-processing), 5.3, 5.26, 10.4, 10.7, 10.8, 11.274 – 11.276
  REN or RENAME (resident), 5.3, 10.5, 10.12, 10.16, 11.277 – 11.280
  RESTORE (transient), 10.5, 10.9, 10.13, 10.14, 10.16, 11.280 – 11.303
  RMDIR or RD (resident), 5.3, 7.7, 7.14, 10.5, 10.12, 10.14, 11.304 – 11.306
  safety, 10.15
  SEARCH (transient), 10.5, 10.14, 11.306 – 11.309
  SET (resident), 5.3, 10.5, 10.7, 10.8, 10.18, 11.309 – 11.314
  SHIFT (resident, batch-processing), 5.3, 5.26, 10.5, 10.7, 11.314 – 11.317
  SHIP (transient), 5.5, 18.1 – 18.3
  SORT (transient), 5.5, 10.5, 10.9, 11.318 – 11.322
  system preparation, 10.18

Index

  - SYS (transient), 5.5, 10.5, 10.13, 10.14, 10.18, 11.322 – 11.325
  - TIME (resident), 5.3, 10.6, 10.9, 10.10, 11.325 – 11.329
  - TREE (transient), 10.6, 10.9, 10.10, 11.330 – 11.336
  - TYPE (resident), 5.3, 7.10, 10.6, 10.4, 11.337 – 11.341
  - VER (resident), 5.3, 10.6, 10.9, 10.11, 11.342
  - VERIFY (resident), 5.3, 10.6, 10.17, 11.343 – 11.344
  - VOL (resident), 5.3, 10.6, 10.9, 10.11, 11.345 – 11.346
- Command scanner with LIB, 13.2
- COMP as a command
  - command line entry of, 11.88 – 11.89
  - comparing two files with, 11.89 – 11.90
  - entry forms of, 11.83
  - error messages of, 11.93 – 11.94
  - help screen of, 11.86
  - interactive entry prompts and responses of, 11.86 – 11.88
  - preliminary concepts of, 11.83 – 11.85
  - purpose of, 11.83
  - wildcard characters with, 11.90 – 11.93
- Compare with DEBUG, 15.5, 15.12
- Component. *See* System Component
- Concatenating files with COPY, 11.120 – 11.122
- COFIG.SYS file
  - creating or changing a, 9.20 – 9.22
  - error message of, 9.23
  - example in, 9.22
- CONFIGUR as a command, 10.2, 10.18
  - advanced concepts of, 11.112 – 11.113
  - configuring a COM device with, 11.104 – 11.109
  - configuring an LPT device with, 11.101 – 11.104
  - entry form of, 11.95
  - error messages of, 11.113
  - main menu of, 11.98 – 11.101
  - preliminary concepts of, 11.95 – 11.98
  - purpose of, 11.95
  - remapping an LPT device with, 11.109 – 11.112
- Control-break. *See* CTRL-BREAK
- Control character entries, 5.20 – 5.21, 5.48
- Controller card of the Winchester, 18.2, 18.3
- Convenience commands, 10.8 – 10.9
- COPY as a command, 7.8, 7.10, 10.3, 10.12, 10.13, 10.16
  - command line entry of, 11.114 – 11.117
  - concatenating files with, 11.120 – 11.122
  - copying a file to another directory with, 11.119 – 11.120
  - copying a file to another disk with, 11.117 – 11.118
  - copying a file to or from a device with, 11.122 – 11.123
  - copying and renaming files with, 11.118 – 11.119
  - entry forms of, 11.114
  - error messages of, 11.123 – 11.126
  - purpose of, 11.114
  - with resident commands, 7.10
  - switch with, 11.114, 11.117

Index

COPYALL as an intraline editing function, 5.17, 12.32, 12.35
COPYUP as an intraline editing function, 5.17, 12.32, 12.34
COPY1 as an intraline editing function, 5.17, 12.32, 12.33 – 12.34
CTRL-BREAK
    to abort a link session, 14.16
    for command interruption, 5.38, 5.49
    as a LIB command character, 13.7, 13.8, 13.18
CTTY as a command, 10.3, 10.18
    advanced concepts of, 11.128
    command line entry of, 11.127
    entry form of, 11.126
    preliminary concepts of, 11.127
    purpose of, 11.126
Current character with the COPY1 function, 12.33
Current working directory, 7.5
Cylinder defined, 16.4

D

d: defined, 5.10, 5.48
Data filtering commands, 10.9
Data/software analysis commands, 10.10 – 10.11
Data/software change commands, 10.11 – 10.12
Data/software creation commands, 10.12 – 10.13
Data/software movement commands, 10.13 – 10.14
DATE as a command, 10.3, 10.8, 10.10
    command line entry of, 11.130 – 11.131
    entry forms of, 11.128
    error message of, 11.132
    interactive entry prompt and response of, 11.129 – 11.130
    preliminary concepts of, 11.128 – 11.129
    purpose of, 11.128
    usage of, 11.131
Date stamp defined, 11.287
DEBUG as a command
    entry form of, 15.1
    error messages of, 15.41
    functions of, 15.4 – 15.5, 15.9 – 15.40
    preliminary concepts of, 15.2 – 15.8
    purpose of, 15.1
Decrypted files with CIPHER, 11.73 – 11.74
Default drive defined, 1.4
Default system prompt defined, 1.4, 5.6
DEL or ERASE as a command, 7.5, 7.10, 10.3, 10.11, 10.16
    command line entry of, 11.133
    deleting files from another disk, 11.134
    deleting files from the default disk, 11.133
    entry forms of, 11.132
    error messages of, 11.135 – 11.136
    purpose of, 11.132
    with resident commands, 7.10
    wildcard characters with, 11.134 – 11.135

Index

DELETE LINES as an EDLIN interline command, 12.5, 12.7 – 12.10
Deleting files with DEL or ERASE, 11.132 – 11.136
Delimiters
  in command lines, 5.11
  defined, 5.7
Destination drive, 5.12
Destination file, 5.12
  with COPY, 11.115 – 11.117
DETECT as a command
  entry form of, 19.1
  error messages of, 19.7 – 19.8
  preliminary concepts of, 19.1 – 19.7
  purpose of, 19.1
Device drivers
  ANSI.SYS as, 11.2 – 11.10
  character, 8.2
  defined, 8.1
  MDISK.DVD as, 11.212 – 11.217
Devices defined, 8.1
DIR as a command, 7.2, 7.10, 10.3, 10.10
  command line entry of, 11.138 – 11.139
  displaying all directory entries with, 11.140 – 11.141
  displaying selected files with, 11.141 – 11.142
  displaying subdirectories of the root directory with, 11.142
  entry forms of, 11.137
  error messages of, 11.144 – 11.145
  with the page mode switch, 11.137, 11.139, 11.143
  printing the directory display with, 11.144
  purpose of, 11.137
  with resident commands, 7.10
  with the wide display mode switch, 11.137, 11.139, 11.143 – 11.144
Directory
  changing a working, 7.12 – 7.13
  commands, 7.7 – 7.8, 10.14
  creating a subdirectory with a working, 7.13
  deleting a, 7.14
  displaying a working, 7.11 – 7.12
  file attributes in a, 7.19
  model, 7.15 – 7.18
Directory display
  of all entries, 11.140 – 11.141
  printing the, 11.144
  of selected files, 11.141 – 11.142
  of subdirectories of the root directory, 11.142
Directory listing, sorting of, 11.320 – 11.321
Directory structure. *See* Hierarchical directory structure
Disk
  comparison of, 11.145 – 11.149
  copy-protected, 3.1

Index

Disk (continued)
    creating exact duplicate of, 11.150 – 11.155
    error, for command interruption, 5.38
    floppy, 1.6 – 1.8
    general, backup procedure for, 3.4 – 3.8
    MS-DOS distribution, backup procedure for, 3.1 – 3.3
    recovering usable portions of, 11.268 – 11.274
DISKCOMP as a command, 10.3, 10.10, 10.17
    advanced concepts of, 11.148
    command line entry of, 11.147 – 11.148
    entry form of, 11.145
    error messages of, 11.149
    interactive entry prompts and responses of, 11.146 – 11.147
    preliminary concepts of, 11.145
    purpose of, 11.145
DISKCOPY as a command, 10.3, 10.12, 10.13, 10.16
    for backing up disks, 3.3, 3.5 – 3.8
    command line entry of, 11.151 – 11.154
    entry form of, 11.150
    error messages of, 11.154 – 11.155
    preliminary concepts of, 11.150
    purpose of, 11.150
    switch with, 11.150, 11.151, 11.153, 11.154
Disk directory with FORMAT, 11.196 – 11.198
Disk drive, 5.12
    defined, 1.3
    single, instructions for, 5.36 – 5.38
Dot (.) as a directory name, 7.5
Dot dot (..) as a directory name, 7.5
Drive name
    defined, 1.4
    as a parameter with DIR, 11.138
    with TYPE, 11.339
Dump with DEBUG, 15.13 – 15.14
Dynamic bootup parameter defaults defined, 6.10 – 6.11

E
ECHO as a command, 10.3, 10.7
    command line entry of, 11.157
    displaying messages during batch file execution with, 11.158 – 11.159
    displaying the status of, 11.159
    entry forms of, 11.156
    preliminary concepts of, 11.156 – 11.157
    purpose of, 11.156
    turning off the command display with, 11.157 – 11.158
Editing functions, special (intraline), 5.17
Editing keys, special, 5.17, 5.48
EDIT LINE as an EDLIN interline command, 12.5, 12.10 – 12.12, 12.31
EDLIN as a command
    command entry form of, 12.1

Index

  error messages of, 12.43 – 12.46
  interline commands of, 12.2, 12.3, 12.4 – 12.30
  intraline editing functions with, 12.2, 12.3, 12.31 – 12.42
  invoking, 12.2 – 12.4
  preliminary concepts of, 12.1 – 12.2
Encrypted files. *See* CIPHER as a command
END EDIT as an EDLIN interline command, 12.3, 12.5, 12.12 – 12.13, 12.21
Enter with DEBUG, 15.15 – 15.17
Entry form notation, xxii-xxv, xxvi-xxviii
Epson MX-80 printer with PSC, 11.261 – 11.262
Equal sign (=) as a delimiter, 5.7, 5.11
ERASE. *See* DEL or ERASE as a command
Error messages
  of APPLY, 11.16
  of ASSIGN, 11.21 – 11.22
  of BACKUP, 11.48 – 11.52
  of Bootup, 6.15 – 6.18
  of CHDIR, 11.58
  of CHKDSK, 11.63 – 11.67
  of CIPHER, 11.74
  of COMMAND, 11.80 – 11.82
  of COMP, 11.93 – 11.94
  of CONFIGUR, 11.113
  of COPY, 11.123 – 11.126
  of DATE, 11.132
  of DEBUG, 15.41
  of DEL or ERASE, 11.135 – 11.136
  of DETECT, 19.7 – 19.8
  of DIR, 11.144 – 11.145
  of DISKCOMP, 11.149
  of DISKCOPY, 11.154 – 11.155
  of EDLIN, 12.43 – 12.46
  of EXE2BIN, 11.161 – 11.162
  of FC, 11.175
  of FIND, 11.182
  of FOR, 11.186
  of FORMAT, 11.198 – 11.199
  of GOTO, 11.201
  of LIB, 13.18 – 13.21
  of LINK, 14.24 – 14.28
  of MAP, 11.211
  of MDISK.DVD, 11.216 – 11.217
  of MKDIR, 11.220 – 11.221
  of MODE, 11.238
  of MORE, 11.241
  operating system, 5.38 – 5.47
  of PART, 17.21 – 17.22
  of PATH, 11.244
  of PREP, 16.13 – 16.14

Index

PART as a command (continued)
    invoking, 17.2 – 17.5
    partitioning menu with, 17.2, 17.5 – 17.8
    partition table with, 17.18 – 17.21
    selecting a boot partition in, 17.8 – 17.10
Partitioning menu, 17.5 – 17.8
    figures of, 17.4, 17.5
Partition. *See also* Working partitions
    allocating a, 17.10 – 17.13
    assignment of a, 11.16 – 11.22
    defined, 1.9, 16.3
    deleting a, 17.15 – 17.17
    error conditions of a, 17.13 – 17.15
    selecting a default boot, 17.8 – 17.10
    with Winchester disk, 1.9 – 1.10, 1.24, 17.1
Partition table
    with PART, 17.18 – 17.21
    with PREP, 16.9 – 16.11
    with Winchester disks,
PATH as a command, 7.7, 7.8 – 7.8, 10.4, 10.12, 10.14
    advanced concepts of, 11.244
    command line entry of, 11.243 – 11.244
    entry form of, 11.241 – 11.242
    error message of, 11.244
    preliminary concepts of, 11.242 – 11.243
    purpose of, 11.241
    with transient commands, 7.8 – 7.9
Path name
    absolute and relative, 7.6
    defined, 7.6
Paths
    and resident commands, 7.10 – 7.11
    and transient commands, 7.8 – 7.9
PAUSE as a command, 10.4, 10.7
    command line entry of, 11.245 – 11.246
    creating a message or prompt with, 11.247
    entry form of, 11.245
    preliminary concepts of, 11.245
    purpose of, 11.245
    use of, 11.246
Period (.) as an EDLIN interline command parameter, 12.5
Peripherals defined, 8.1
Physical components, 9.4
    disk locations of, 9.9 – 9.11
    memory locations of, 9.12
    of MS-DOS, 9.5 – 9.9
Pipeline defined, 8.12
Pipe (|) with MORE, 11.239 – 11.240
Pipes
    defined, 8.12
    with input/output, 8.11 – 8.14

Platters defined, 16.2
Plus sign (+)
  as a delimiter, 5.7, 5.11
  as a LIB command character, 13.1, 13.7, 13.8, 13.9 – 13.10, 13.14, 13.15
  with LINK, 14.15 – 14.16
Pound sign (#) as an EDLIN interline command parameter, 12.6
PREP as a command, 16.1
  boot indicator byte with, 16.9
  boot record with, 16.8
  initializing disks with, 16.5 – 16.6
  initializing reserved Winchester area with, 16.7
  operation of, 16.5 – 16.7
  testing media with, 16.6
PRINT as a command, 10.4, 10.15
  command line entry of, 11.249 – 11.252
  displaying contents of print queue with, 11.254 – 11.255
  entry forms of, 11.248
  error messages of, 11.256 – 11.258
  manipulating files in print queue with, 11.255 – 11.256
  preliminary concepts of, 11.248 – 11.249
  printing a series of files with, 11.253 – 11.254
  purpose of, 11.247 – 11.248
  switches with, 11.248, 11.250 – 11.252
Printek 920 printer with PSC, 11.261
Printer echo
  ASCII screen displays with, 8.16
  invoking, 8.15
  printing screen displays with, 8.17
Printer options with PSC, 11.261, 11.262 – 11.263
Print queue
  displaying, 11.254 – 11.255
  manipulating files in, 11.255 – 11.256
PROMPT as a command, 10.4, 10.8
  advanced concepts of, 11.261
  command line entry of, 11.259 – 11.260
  entry form of, 11.258
  preliminary concepts of, 11.259
  purpose of, 11.258
Protocol defined, 11.95
PSC as a command, 10.5, 10.15
  command line entry of, 11.262 – 11.263
  entry form of, 11.261
  graphics or special characters, screen displays with, 8.17 – 8.18, 11.261
  preliminary concepts of, 11.262
  purpose of, 11.261
  selective screen displays with, 8.16

Index

Q

Question mark (?)
  with ASSIGN, 11.18
  with EDLIN, 12.6
  as a wildcard character, 1.15
Quit with DEBUG, 15.29
QUIT EDIT as an EDLIN interline command, 12.3, 12.5, 12.21 – 12.22
QUIT INPUT or VOID as an intraline editing function, 5.17, 12.32, 12.37 – 12.38

R

RDCPM as a command, 10.5, 10.13, 10.16
  command line entry of, 11.264 – 11.267
  entry forms of, 11.263 – 11.264
  error message of, 11.268
  preliminary concepts of, 11.264
  purpose of, 11.263
  switch with, 11.264
Read/write heads
  defined, 1.5, 16.2
  of the Winchester disk, 18.1, 18.2, 18.3
RECOVER as a command, 10.5, 10.14, 10.16, 10.17
  command line entry of, 11.269 – 11.271
  entry forms of, 11.268
  error message of, 11.274
  preliminary concepts of, 11.268 – 11.269
  purpose of, 11.268
  recovering an entire disk with, 11.272 – 11.274
  recovering a single file with, 11.271 – 11.272
Register with DEBUG, 15.30 – 15.32
Relative path name, 7.6
Remap defined, 11.109
Remark
  creating a, with PAUSE, 11.245 – 11.246, 11.247
  creating a, with REM, 11.274 – 11.276
REM as a command, 10.5, 10.7, 10.8
  command line entry of, 11.275
  entry form of, 11.274
  preliminary concepts of, 11.275
  purpose of, 11.274
  use of, 11.275 – 11.276
RENAME. *See* REN or RENAME as a command
Renaming files with COPY, 11.118 – 11.119
REN or RENAME as a command, 10.5, 10.12, 10.16
  command line entry of, 11.277 – 11.278
  entry forms of, 11.277
  error message of, 11.280
  purpose of, 11.277
  renaming files on the default disk with, 11.278 – 11.279
  renaming files on a non-default disk with, 11.279
  with wildcards, 11.279 – 11.280

Replaceable parameters (shift), 11.314 – 11.315
REPLACE TEXT as an EDLIN interline command, 12.5, 12.6, 12.22 – 12.26
Resident commands, 1.18, 1.19, 5.2 – 5.4, 5.5, 7.10 – 7.11
  BREAK as, 5.3, 10.2, 10.17, 11.52 – 11.54
  CHDIR or CD as, 5.3, 7.5, 7.7, 7.12, 10.2, 10.10, 10.11, 10.14, 11.54 – 11.58
  CLS as, 5.3, 10.2, 10.7, 11.74 – 11.75
  command line entry of, 5.5, 5.8
  COPY as, 5.3, 7.8, 7.10, 10.3, 10.12, 10.13, 10.16, 11.114 – 11.126
  CTTY as, 5.3, 10.3, 10.18, 11.126 – 11.128
  d:, 5.3
  DATE as, 5.3, 10.3, 10.8, 10.10, 11.128 – 11.132
  DEL or ERASE as, 5.3, 7.5, 7.10, 10.3, 10.11, 10.16, 11.132 – 11.136
  DIR as, 5.3, 5.5, 7.2, 7.10, 10.3, 10.10, 11.137 – 11.145
  ECHO as, 5.3, 5.25, 10.3, 10.7, 11.156 – 11.159
  EXIT as, 5.3, 10.3, 10.8, 11.162 – 11.163
  FOR as, 5.3, 5.25, 10.4, 10.7, 11.183 – 11.186
  GOTO as, 5.3, 5.25, 10.4, 10.7, 11.199 – 11.201
  IF as, 5.3, 5.26, 10.4, 10.7, 11.201 – 11.206
  MKDIR or MD as, 5.3, 7.7, 7.13, 10.4, 10.12, 10.14, 11.218 – 11.221
  PATH as, 5.3, 7.7, 7.8 – 7.9, 10.4, 10.12, 10.14, 11.241 – 11.244
  paths and, 7.10 – 7.11
  PAUSE as, 5.3, 5.26, 10.4, 10.7, 11.245 – 11.247
  PROMPT as, 5.3, 10.4, 10.8, 11.258 – 11.261
  REM as, 5.3, 5.26, 10.4, 10.7, 10.8, 11.274 – 11.276
  REN or RENAME as, 5.3, 10.5, 10.12, 10.16, 11.277 – 11.280
  RMDIR or RD as, 5.3, 7.7, 7.14, 10.5, 10.10, 10.13, 11.304 – 11.306
  SET as, 5.3, 10.5, 10.7, 10.8, 10.18, 11.309 – 11.314
  SHIFT as, 5.3, 5.26, 10.5, 10.7, 11.314 – 11.317
  TIME as, 5.3, 10.6, 10.9, 10.10, 11.325 – 11.329
  TYPE as, 10.6, 10.11, 11.337 – 11.341
  VER as, 5.3, 10.6, 10.9, 10.11, 11.342
  VERIFY as, 5.3, 10.6, 10.17, 11.343 – 11.344
  VOL as, 5.3, 10.6, 10.9, 10.11, 11.345 – 11.346
Response file defined, 13.6
Response file method
  to invoke LIB, 13.7, 13.11 – 13.12
  to invoke LINK, 14.13, 14.18 – 14.19
RESTORE as a command, 10.5, 10.9, 10.13, 10.14, 10.16
  command line entry of, 11.285 – 11.296
  entry forms of, 11.281
  error messages of, 11.300 – 11.303
  help screen of, 11.283 – 11.284

Index

Switches (continued)
  with FIND, 11.176, 11.178 – 11.180
  with FORMAT, 11.186 – 11.187, 11.190, 11.191, 11.193 – 11.19S
  with LINK, 14.22 – 14.24
  with PRINT, 11.248, 11.250 – 11.2S2
  with RDCPM, 11.264
  with RESTORE, 11.281, 11.287 – 11.296
  with SEARCH, 11.307
  with SORT, 11.318, 11.319 – 11.320
Syntax for DEBUG functions, 15.5
SYS as a command, 10.5, 10.13, 10.14, 10.18
  advanced concepts of, 11.324
  command line entry of, 11.323 – 11.324
  entry form of, 11.322
  error messages of, 11.324 – 11.325
  preliminary concepts of, 11.323
  purpose of, 11.322
System call defined, 1.2
System component, 9.1, 9.23 – 9.24
  behavior after bootup of, 9.16 – 9.20
  behavior during bootup of, 9.13 – 9.16
  disk locations of, 9.9 – 9.11
  functional, 9.1 – 9.4
  memory location of, 9.12
  physical, 9.4 – 9.9
System environment with SET, 11.309, 11.310, 11.311, 11.312 – 11.313
System memory defined, 6.2
System preparation commands, 10.18
System prompt
  with COMMAND.COM, S.3, S.4
  of a command line, S.7
  defined, 1.3

T

Tab as a delimiter, 5.7
Template
  defined, 5.14
  with intraline editing, 12.31, 12.32, 12.40 – 12.42
TIME as a command, 10.6, 10.9, 10.10
  command line entry of, 11.328
  entry forms of, 11.325 – 11.326
  error message of, 11.329
  interactive entry prompt and response of, 11.326 – 11.328
  preliminary concepts of, 11.326
  purpose of, 11.325
  use of, 11.329
Trace with DEBUG, 15.34 – 1S.3S
Tracks defined, 1.5, 16.4
Transfer system files with SYS, 11.322 – 11.32S
Transient commands, 1.18, 1.19 – 1.20, 5.2, 5.4 – 5.5, S.6, 7.8 – 7.9

  APPLY as, 10.2, 10.15, 11.11 – 11.16
  ASSIGN as, 10.2, 10.11, 11.16 – 11.22
  BACKUP as, 10.2, 10.10, 10.12, 10.13, 10.15, 11.23 – 11.52
  CHKDSK as, 5.4, 10.2, 10.10, 10.11, 10.17, 11.58 – 11.67
  CIPHER as, 10.2, 10.9, 10.16, 11.67 – 11.74
  COMMAND as, 5.4, 10.2, 10.8, 11.75 – 11.82
  command line entry of, 5.5, 5.8 – 5.9
  COMP as, 10.2, 10.10, 10.17, 11.83 – 11.94
  CONFIGUR as, 5.4, 10.2, 10.18, 11.95 – 11.113
  DEBUG as, 5.4, 15.1 – 15.41
  DETECT as, 5.4, 19.1 – 19.8
  DISKCOMP as, 5.4, 10.3, 10.10, 10.17, 11.145 – 11.149
  DISKCOPY as, 3.3, 3.5 – 3.8, 5.4, 10.3, 10.12, 10.13, 10.16, 11.150 – 11.155
  EDLIN as 5.4, 12.1 – 12.46
  EXE2BIN as, 11.159 – 11.162
  FC as, 5.5, 10.3, 10.10, 10.17, 11.163 – 11.175
  FIND as, 5.5, 10.3, 10.9, 11.176 – 11.182
  FORMAT as, 5.5, 10.4, 10.11, 10.12, 10.14, 10.18, 11.186 – 11.199
  LIB as, 5.4, 13.1 – 13.21
  LINK as, 5.4, 14.1 – 14.28
  MAP as, 10.4, 10.8, 10.11, 11.207 – 11.211
  MODE as, 10.4, 10.18, 11.221 – 11.238
  MORE as, 5.4, 10.4, 10.9, 11.239 – 11.241
  PART as, 5.4, 17.1 – 17.22
  paths and, 7.8 – 7.9
  PREP as, 5.4, 16.1 – 16.14
  PRINT as, 5.4, 10.4, 10.15, 11.247 – 11.258
  PSC as, 10.4, 10.15, 11.261 – 11.263
  RDCPM as, 5.4, 10.4, 10.13, 10.16, 11.263 – 11.268
  RECOVER as, 5.4, 10.4, 10.14, 10.16, 10.17, 11.268 – 11.274
  RESTORE as, 10.5, 10.9, 10.13, 10.14, 10.16, 11.280 – 11.303
  SEARCH as, 10.5, 10.14, 11.306 – 11.309
  SHIP as, 5.5, 18.1 – 18.3
  SORT as, 5.5, 10.5, 10.9, 11.318 – 11.322
  SYS as, 5.5, 10.5, 10.13, 10.14, 10.18, 11.322 – 11.325
  TREE as, 10.6, 10.9, 10.10, 11.330 – 11.336
Transtar 315 color printer with PSC, 11.261
TREE as a command
  command line entry of, 11.333
  entry forms of, 11.330
  error messages of, 11.336
  examples of, 11.334 – 11.336
  help screen of, 11.332
  preliminary concepts of, 11.330 – 11.332
  purpose of, 11.330
TYPE as a command, 7.10, 10.6, 10.11
  command line entry of, 11.339 – 11.340
  displaying a file on the default disk with, 11.340

Index

TYPE as a command (continued)
    displaying a file on a non-default disk with, 11.340 – 11.341
    entry forms of, 11.337
    error messages of, 11.341
    preliminary concepts of, 11.338 – 11.339
    purpose of, 11.337
    with resident commands, 7.10

U

Unassemble with DEBUG, 15.36 – 15.38
Utilities, 5.4 – 5.5. *See also* Transient commands

V

VER as a command, 10.6, 10.9, 10.11
    command line entry of, 11.342
    entry form of, 11.342
    purpose of, 11.342
VERIFY as a command, 10.6, 10.17
    advanced concepts of, 11.344
    command line entry of, 11.344
    entry form of, 11.343
    preliminary concepts of, 11.343 – 11.344
    purpose of, 11.343
VM.TMP temporary file used with LINK, 14.12
VOID. *See* QUIT INPUT or VOID as an intraline editing function
VOL as a command, 10.6, 10.9, 10.11
    command line entry of, 11.345 – 11.346
    entry form of, 11.345
    preliminary concepts of, 11.345
    purpose of, 11.345
Volume defined, 11.24

W

Wildcard characters
    defined, 1.15
    to reference multiple files, 1.14 – 1.17
Winchester disk drive
    and disks, 1.9 – 1.10
    features of, 16.2 – 16.3
    initializing reserved area of, 16.7
    making working partitions from, 4.1, 4.11 – 4.20
    with RDCPM, 11.264
    reserved area of, 16.7 – 16.13
    with SHIP, 18.1 – 18.3
    tracks and cylinders of, 16.4 – 16.5
Winchester partitions
    with ASSIGN, 11.16 – 11.22
    isolating bad sectors of, 19.1 – 19.8
Working disks, 1.9 – 1.10
    CONFIGUR with, 4.8 – 4.10

Index

- COPY with, 4.6 – 4.7
- FORMAT with, 4.4 – 4.6
- procedure for, 4.1, 4.2 – 4.10
- requirements for, 4.2
- startup activity with, 4.3

Working partitions
- ASSIGN with, 4.12 – 4.13
- CONFIGUR with, 4.16 – 4.18
- COPY with, 4.15 – 4.16
- floppy disk startup with, 4.12
- FORMAT with, 4.14 – 4.15
- procedure for, 4.1, 4.11 – 4.20
- requirements for, 4.11
- startup activity with, 4.19 – 4.20

Write with DEBUG, 15.39 – 15.40

WRITE LINES as an EDLIN interline command, 12.3, 12.5, 12.6, 12.7, 12.30

Zenith | data systems HEATH

IMPORTANT NOTICE

This notice was written to give you a brief explanation of three new features of MS-DOS version 2. They are not documented in the MS-DOS manual. Accompanying this notice is detailed documentation on the use of two of these features—the CONFIGUR Automatic Partition Assignment Option and the KEYB*xxxx* command. You will be instructed later in this notice where to insert the documention into the manual. The third new feature, the Hewlett Packard ThinkJet Printer file, is explained in the section below.

Since KEYB*xxxx* is a new command it is not listed in your table of contents. Therefore, locate Chapter 11 in the table of contents. After the IF command write in KEYB*xxxx* ...... 11.206a.

PSCHPTJ.COM File

MS-DOS version 2 now contains a print screen file for the Hewlett Packard ThinkJet Printer. The PSC (print screen) commands are explained in detail in Chapter 11. **PSCHPTJ** is the command entry for this printer.

CONFIGUR Automatic Partition Assignment Option

There is a total of four drive letters you can assign to your MS-DOS partition(s). When booting up on an MS-DOS partition, that partition is automatically assigned the first available drive letter. If you have other MS-DOS partitions, use the ASSIGN command to designate them a drive letter. (Refer to the ASSIGN command in Chapter 11 for details.) However, most users do not assign every one of the four drive letters that are available. These available-but-unassigned drive letters present a problem for some applications such as Topview and Microsoft Project. The problem occurs when the application program attempts to determine the number of Winchester drives by examining each possible drive letter. MS-DOS tells the application program that there are four possible drives, so the program expects to find all four. However, the program finds only the drives that have been assigned. Therefore, you receive the following error message:

```
Not ready error reading drive x
Abort, Retry, Ignore?
```

To correct this problem, an option has been added to the CONFIGUR command. It allows you to assign your MS-DOS partition(s) manually as described above or have them assigned automatically. Details on how to use this option are contained in the accompanying pages, which are to replace existing pages in Chapter 11.

Below is a list of the existing pages to be discarded and what pages replace them.

EXISTING PAGE(S)	REPLACEMENT PAGE(S)
11.95—11.101	11.95—11.101b
11.104	11.104
11.110	11.110
11.112	11.112—11.112b

NOTE: These page numbers refer to the currently shipping manual. They may be slightly different in older versions of the manual.

KEYB*XXXX* Command

This command allows you to load a foreign language keyboard layout into memory. Details on how to use this command are in the accompanying pages, which must be inserted in your manual. Insert pages 11.206a, 11.206b, and 11.206c after the IF command in Chapter 11.

Thank you,

Zenith Data Systems

CONFIGUR (Transient)

Purpose

The CONFIGUR program configures the system to use the correct protocol for your printer, modem or other serial/parallel device.

Entry Form

```
CONFIGUR
```

Preliminary Concepts

CONFIGUR is a utility program which allows you to configure your system's logical devices so that they fit the protocol requirements of your printer, modem or other serial/parallel devices that you connect to your microcomputer. *Protocol* is the word used when referring to the method of communications required for data transfer between a microcomputer and a peripheral. The protocol also offers a means of controlling how the peripheral performs its specific operations. A device's protocol consists of a set of standard signals that the hardware device recognizes and uses so that its operation may be correctly synchronized and interpreted.

NOTE: If you are using a parallel printer only, connect it to parallel port number 1. This port is already configured for LPT1, so you do not need to run the CONFIGUR program.

CONFIGUR gives you a quick way to select the protocol that your devices can recognize. CONFIGUR enables the Input/Output Manager (IO.SYS) to communicate with your devices using the correct baud rate, communication lines, control and data signals necessary for information transfer to a particular peripheral device.

CONFIGUR

Since CONFIGUR adjusts your system for hardware characteristics, you must answer prompts that ask about these characteristics. Always consult your hardware manuals and check the settings of your hardware devices when answering these prompts.

CONFIGUR is a *menu-driven* command. This means that your screen will display a list, or menu, of selections from which you may choose. The first menu, referred to as the *main menu*, contains choices that will lead you to a series of submenus. Many of these submenus will list protocol characteristics for peripheral devices. You should refer to the manual for a specific peripheral when selecting values from these menus to make sure you are selecting the proper values.

To use the menu, press the letter that corresponds to the menu selection you wish to choose.

The configurable logical devices are the PRN (LPT1, 2, and 3) and the AUX (COM1 and 2) devices. A logical device name is assigned to a table of information that is contained in IO.SYS. When that device name is used in a command to read or write data, IO.SYS uses the information it has about that device to carry out the command.

These devices are named to the system in the following way:

Table 11.1. System Device Names

DEVICE NAME	DEVICE FUNCTION
PRN or LPT1	Parallel printer device #1
AUX or COM1	Serial device #1
COM2	Serial device #2
LPT2	Parallel printer device #2
LPT3	Parallel printer device #3

Logical devices configurable to different identities make it easy for the operating system to use many devices. If the operating system could not use a logical device to represent a physical device, it would either have to be rewritten every time a different device was used, or it would have to contain information about *all* possible devices.

Logical devices present a very practical approach to enable the operating system to handle many different devices. When confronted with one of the reserved logical device names (for example LPT1 or COM1), the operating system knows exactly where to look to get information currently used to describe the device.

CONFIGUR also knows exactly where the descriptions of the logical devices are stored. When CONFIGUR is told to save a new description, it overwrites the old device description. This means that if the PRN device is currently the description of an MX-80 printer, commands that have PRN as an argument cause the operating system to use the MX-80 description. If CONFIGUR is then invoked and the PRN device is redefined to the characteristic protocol of a Diablo 1640 printer, the operating system then recognizes and uses the different protocol. Even though the actual physical device changes, MS-DOS is able to communicate to it because it looks up the device's definition under the logical name.

The following discussion of CONFIGUR is divided into four areas, each describing the configuration of a particular kind of device.

These areas include:

1. The main menu. This is a description and explanation of the main menu in CONFIGUR and what the various selections mean for you.

2. Configuring an LPT Device. This is a printer-device and uses one of the parallel ports.

NOTE: If you are using a serial printer, follow the instructions given in Remapping an LPT Device of this section.

3. Configuring a COM Device. This is a serial device and includes such peripherals as modems and some printers.

4. Remapping an LPT Device. This is typically used to configure your system for a serial printer, such as the MX80-Serial. To accomplish this, you must use the following procedure:

   A. Configure a COM device (see 3, above) for your particular printer.

   B. Remap LPT1 to the COM device you configured for your printer. Make sure the COM devices are the same. For example, if you configure COM1 for your printer—you must remap LPT1 to COM1, not COM2.

   C. You must connect your printer to the configured COM port using a *null modem* cable. Do not use any other type of cable.

The Main Menu

To begin to configure your system for a particular device's protocol, enter

```
CONFIGUR
```

and press **RETURN**. CONFIGUR will display a menu of choices for configuring your system. Since CONFIGUR checks your hardware configuration when it is invoked, it knows what ports are connected to your system and will only display valid choices. For example, if you did not have any serial or parallel ports connected to your microcomputer and you invoked CONFIGUR, it would display the message:

```
A>CONFIGUR

No serial or parallel devices are attached.
No configuration is possible.
```

Now, examine the menu that would appear if you have a standard configuration (a system with one parallel port and two serial ports). In this situation, when you enter

```
A>CONFIGUR
```

and press **RETURN**, the screen will display:

```
                       CONFIGUR Version 2.0


Use one of the following options to configure a device

A.  Configure LPT device
B.  Configure COM device

C.  Exit with no changes

Enter selection (A-C)
```

At this main menu, you press in the letter A, B, or C (corresponding to the choice you want to make).

NOTE: The MS-DOS version numbers are shown in the screen displays as the variable, 2.0. This is because your screen display may be a more recent version than that available when this manual was prepared.

Press C to return to the system prompt without making any configuration changes.

Depending on the selection you choose from this main menu, additional menus may appear that offer alternate system configuration options.

Because there are several menus and options that are available, depending on the options you select, ending the program is discussed first before describing the various configurations that are available.

CONFIGUR

After making your selections, you will return to this main menu (notice that three additional options appear):

```
                    CONFIGUR Version 2.0


Use one of the following options to configure a device

A.  Configure LPT device
B.  Configure COM device

Use one of the following to modify an existing system

C.  Exit program
D.  Make changes to disk
E.  Make changes to memory
F.  Make changes to both disk and memory

Enter selection (A-F):
```

Options D, E, and F will appear on the main menu after you have made a configuration. These options allow you to save the configuration that you have selected.

You may still exit without saving any of the options that you have selected by pressing **C**.

Option D saves to disk the changes that you selected. After selecting option **D**, you are asked which disk you wish the changes saved to:

```
Enter drive name with system to modify (A-x):
```

You may save the configuration you selected to any valid drive in your system, hence the *x*. The *x* represents the disk drive having the greatest alphabetic drive name in your particular system.

This feature makes it possible to configure the system on disks other than your current system disk. This can be a useful feature if you have other disks that will be used with different devices for special purposes. This means you can change your current system disk's configuration without disturbing the system configuration that is currently in memory at the time you use CONFIGUR.

Selecting option **E**, alters the system configuration that is currently in your microcomputer's memory, without making that configuration permanent by saving it to a disk. When you press **CTRL-ALT-DEL** and reboot the operating system, the system configuration you selected will be erased and the one on the disk is again read into memory. This allows you to make a temporary change.

NOTE: Changes specified through the CONFIGUR utility can only be recorded on a disk if the disk is *write-enabled*. If you are performing the CONFIGUR operation with a *write-protected* disk, you can only apply changes to the system in memory.

Select the **F** option if you want to make the changes to both memory and disk. This option also asks you for the drive name where you want to save the changes. Therefore, you can change memory and disks other than your booted disk, if you wish.

From the main menu, options A and B lead to sets of secondary menus that look similar.

Configuring an LPT device

If you choose option A, Configure LPT device, from the main menu, the following submenu will appear:

```
Use one of the following options to select the type of configuration

A. Map parallel output to serial output
B. Configure parallel device

C. Exit

Enter selection (A-C):
```

If you wish to configure a parallel device, you should now enter option **B**, to configure the parallel device. (Mapping parallel output to serial output is covered in a later section.) Selecting option B will display the following menu:

CONFIGUR

```
Select the parallel device to be configured

A. LPT1

B. Exit

Enter selection (A-B):
```

NOTE: In this example there is one LPT device, LPT1. Your system may have one, two, or three LPT devices. The menu will reflect your particular system.

If you choose to configure the LPT1 parallel device, by entering option **A**, the following prompts will appear sequentially:

```
Answer the following questions with Y for Yes and N for No

Strip parity on output? (Y/N) <N>
Map lower case to upper case on output? (Y/N) <N>
```

NOTE: In the previous questions, the answers in angle brackets (<>) are the defaults and can be entered by pressing RETURN.

If you wish to strip parity on output (set the eighth bit of each character being sent to the LPT device to 0), press **Y**. If you wish all lowercase characters to appear as uppercase, press Y. After you have answered the questions, the following series of questions will be displayed:

```
If you do not wish a pad character, simply
 press the RETURN key, and then enter a zero
 as the number of pad characters, otherwise press
 the actual key character you wish to pad

For example, to pad after all carriage returns, press
 the RETURN key.

Press the key corresponding to your desired pad character:
Enter the number of pad characters to send (0-255):
```

NOTE: The option to insert pad characters is provided for older, and hence, slower, printer devices which may not be able to react as quickly to the signals coming from your microcomputer. Usually pad characters are sent after a RETURN to allow time for the print element in your printer to return to column 1. It may need this extra time if it is of the older variety.

```
The timeout value is used to give slow devices time to respond to
Input/Output requests. A small value is usually sufficient, but a
number from 0 to 255 can be entered.


Enter timeout value for LPT1:
```

NOTE: The timeout value is provided for reasons roughly similar to the pad characters. However, a timeout value can be set for any type of device, not just printers. The timeout value is a number from 0 to 255 that represents the number of "cycles" that your microcomputer should wait for a device to respond, before it stops trying to communicate. The slower the device you are configuring your microcomputer for, the larger the number you should enter for the timeout value.

After you have entered a timeout value (even if it is zero), you will be returned to the following menu:

```
Use one of the following options to select the type of configuration

A. Map parallel output to serial output
B. Configure parallel device

C. Exit

Enter selection (A-C):
```

If you have configured everything that you need, press C and exit. This will return you to the main menu shown again on the next page:

```
                   CONFIGUR Version 2.0


Use one of the following options to configure a device

A.  Configure LPT device
B.  Configure COM device


   Use one of the following to modify an existing system

C.  Exit program
D.  Make changes to disk
E.  Make changes to memory
F.  Make changes to both disk and memory

Enter selection (A-F):
```

If you have completed the necessary configuration for your system, then you should now save the changes you have made to either disk, or memory, or both.

If you also need to configure a COM device, do not save the changes you have already made at this time, but select option **B**, Configure COM device.

Configuring a COM Device

If you choose option B, Configure COM device, from the main menu, the following menu will be displayed:

```
Select the serial port to be configured

A. COM1
B. COM2

C. Exit

Enter selection (A-C):_
```

NOTE: Two COM devices are shown in this example. If your system has only one COM device, only COM1 will be displayed in this menu.

NOTE: The following prompt will only be displayed if option D, E, F, G, or H was selected from the handshake menu.

```
The timeout value is used to give slow devices time to respond to
Input/Output requests. A small value is usually sufficient, but a
number from 0 to 255 can be entered.

Enter timeout value for COM1:
```

NOTE: The timeout value is provided for reasons roughly similar to the pad characters. However, a timeout value can be set for any type of device, not just printers. The timeout value is a number from 0 to 255 that represents the number of "cycles" that your microcomputer should wait for a device to respond, before it stops trying to communicate. The slower the device you are configuring your microcomputer for, the larger the number you should enter for the timeout value.

Enter a timeout value, even if it is zero.

After you have finished entering the answers to these prompts, you will automatically be returned to the main menu. When the main menu displays, you will have the choice of exiting the program without making any changes; or making changes to disk, memory, or both.

Remapping an LPT Device

If you wish to use a printer with a serial interface you not only need to configure a COM device for that printer, but you need to logically connect, or remap the parallel output to the serial output of that COM device. This step is absolutely necessary if you plan to use a serial printer with your microcomputer. The procedure for remapping is described below. To begin, enter

```
A>CONFIGUR
```

and press **RETURN**; the screen will display

```
                    CONFIGUR Version 2.0


Use one of the following options to configure a device

A.  Configure LPT device
B.  Configure COM device

C.  Exit with no changes

Enter selection (A-C)
```

At this main menu, you should press the letter **A**, and the LPT menu will display.

```
Use one of the following options to select the type of configuration

A. Map parallel output to serial output
B. Configure parallel device

C. Exit

Enter selection (A-C):
```

Choose option A from this menu.

NOTE: If you are going to use a serial printer, you must connect it to the COM1 or COM2 port with a *null modem* cable, available through your dealer.

After A is entered, the following menu is displayed:

```
Select the parallel port to be mapped.

A. LPT1
B. LPT2
C. LPT3

D. Exit

Enter selection (A-D):
```

NOTE: Your system may have only one or two parallel ports, however, this menu will always be shown with all three LPT devices. This is because they represent logical devices, as opposed to physical devices, and can always be remapped.

If you choose **A**, the following menu is displayed:

```
Use one of the following to select a mapping for LPT1

A. No mapping
B. Map to COM1
C. Map to COM2

Enter one of the mapping values:
```

Choose option B or C, depending on which COM port you want the LPT device to be mapped—that is, to which COM port you wish the printer connected.

Option A (No Mapping) is used to undo a previous remapping or *unmap*. This returns the system to the default values.

Whatever option you select at this point will place you back in the menu two levels previous to the last one, as shown:

```
Use one of the following options to select the type of configuration

A. Map parallel output to serial output
B. Configure parallel device

C. Exit

Enter selection (A-C):
```

If you have configured everything that you need, press **C**, and exit. This will bring you back to the main menu shown again on the next page:

```
                     CONFIGUR Version 2.0


Use one of the following options to configure a device
A.  Configure LPT device
B.  Configure COM device

Use one of the following to modify an existing system

C.  Exit program
D.  Make changes to disk
E.  Make changes to memory
F.  Make changes to both disk and memory

Enter selection (A-F):
```

CAUTION: If you have not previously configured the COM device to which you mapped the parallel output, you must do so now. When you use a serial printer, you must configure the COM port to be used (either COM1 or COM2) *and* remap the parallel output to that port. If you do not complete both of these steps, your serial printer will not function.

If everything is complete, you should now save the changes you have made to either disk, or memory, or both.

Advanced Concepts

The CONFIGUR program provides a way for you to change peripherals as your needs change. If you need to save time with a printer, you may choose a high-speed dot-matrix printer, which might use an RTS positive handshake at 4800 baud. Or, your printer needs might require a cleaner, crisper look for correspondence and contracts, so you might choose a daisy-wheel printer that uses ETX/ACK handshaking and operates at 1200 baud.

CONFIGUR makes it possible for you to connect both printers at the same time or alter the configuration quickly so that you are able to use one printer, run CONFIGUR, connect a different printer, and immediately start to use *that* printer.